Computer Vision
Object Deetection LaTeX report

Nouran Hisham 6532
Nourhan Waleed 6609
Kareem Sabra 6594

December 2022

Index

Problem statement

In this assignment, we worked on Cocco dataset which is a large-scale object detection, segmentation, and captioning dataset. We ran 3 different object detectors on this dataset. We learnt to use and differentiate between the architectures. Either PyTorch or TensorFlow are allowed to run our models.

The code for model (1) can be viewed from here

The code for model (2) can be viewed from here

The code for model (3) can be viewed from here

1 Download the dataset

1.1 Dataset format

The Microsoft Common Objects in Context (COCO) dataset is the gold standard benchmark for evaluating the performance of state of the art computer vision models. COCO contains over 330,000 images, of which more than 200,000 are labelled, across dozens of categories of objects. COCO is a collaborative project maintained by computer vision professionals from numerous prestigious institutions, including Google, Caltech, and Georgia Tech.

The COCO dataset is designed to represent a vast array of things that we regularly encounter in everyday life, from vehicles like bikes to animals like dogs to people.
The COCO dataset contains images from over 80 "object" and 91 generic "stuff" categories, which means the dataset can be used for benchmarking general-purpose models more effectively than small-scale datasets.

In addition, the COCO dataset contains:
121,408 images
883,331 object annotations
80 classes of data
The median image ratio is 640 x 480

We worked with COCO dataset 2017, validation split (5000 images) as test split wasn't annotated so we couldn't evaluate our models.

We used FiftyOne to facilitate using the data dute to it's large size. FiftyOne is an open-source tool for building high-quality datasets and computer vision models. The FiftyOne API and App enable you to visualize datasets and interpret models faster and more effectively.

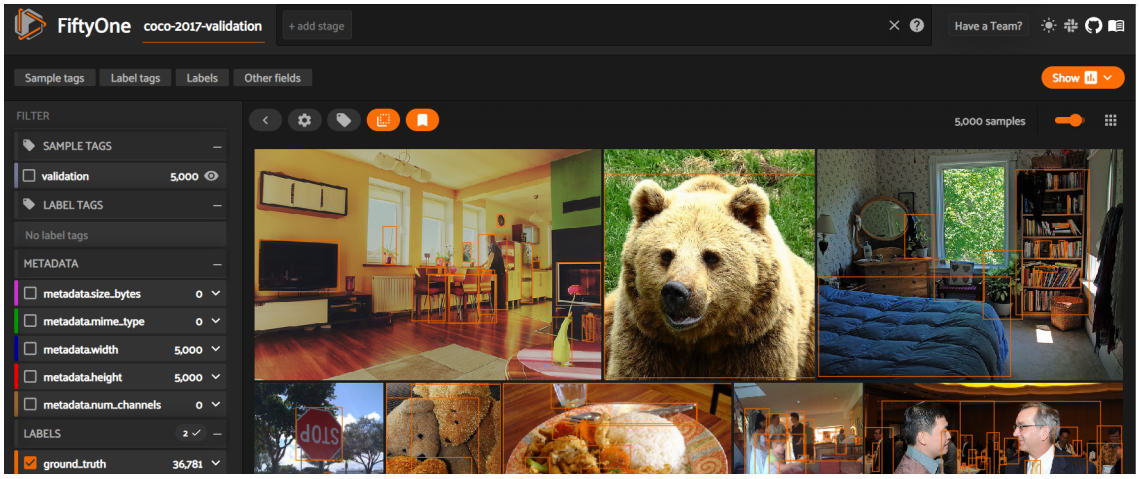
```
[ ] print(dataset)
```

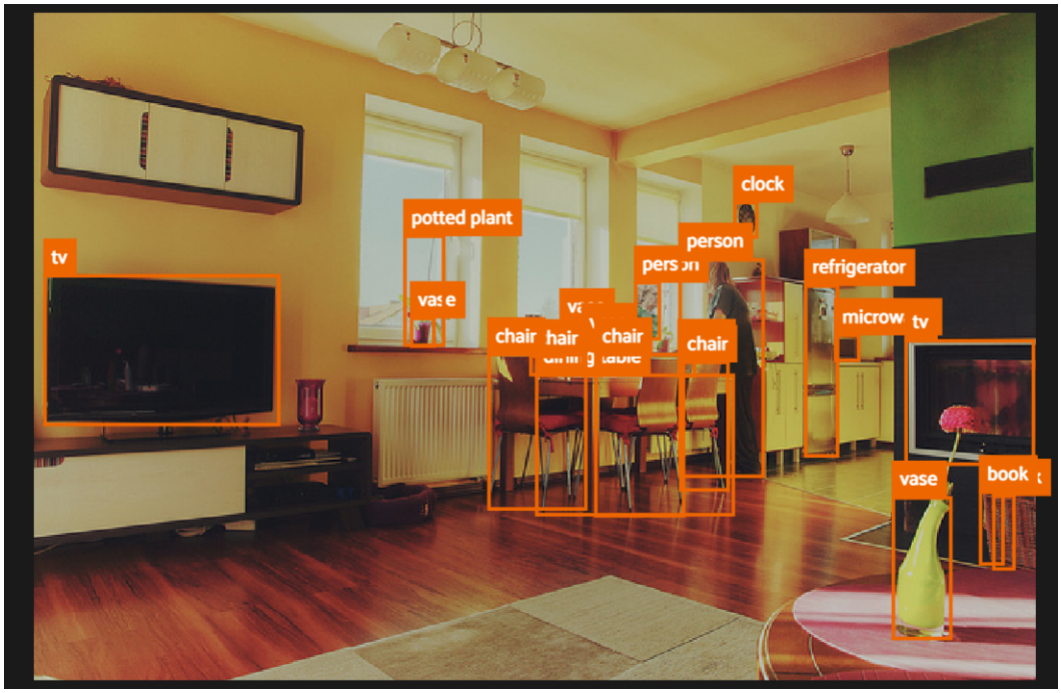
```
Name:        coco-2017-validation
Media type:  image
Num samples: 5000
Persistent:  False
Tags:        []
Sample fields:
    id:                  fiftyone.core.fields.ObjectIdField
    filepath:            fiftyone.core.fields.StringField
    tags:                fiftyone.core.fields.ListField(fiftyone.core.fields.StringField)
    metadata:            fiftyone.core.fields.EmbeddedDocumentField(fiftyone.core.metadata.ImageMetadata)
    ground_truth:        fiftyone.core.fields.EmbeddedDocumentField(fiftyone.core.labels.Detections)
```

```
[ ] sample = dataset.first()
    print(sample.ground_truth.detections[0])
```

```
<Detection: {
    'id': '63a45b63af256b54b23e1c9e',
    'attributes': {},
    'tags': [],
    'label': 'potted plant',
    'bounding_box': [
        0.37028125,
        0.3345305164319249,
        0.038593749999999996,
        0.16314553990610328,
    ],
    'mask': None,
    'confidence': None,
    'index': None,
    'supercategory': 'furniture',
    'iscrowd': 0,
    'eval': 'fn',
    'eval_id': '',
}>
```

1.2 Dataset visualisation



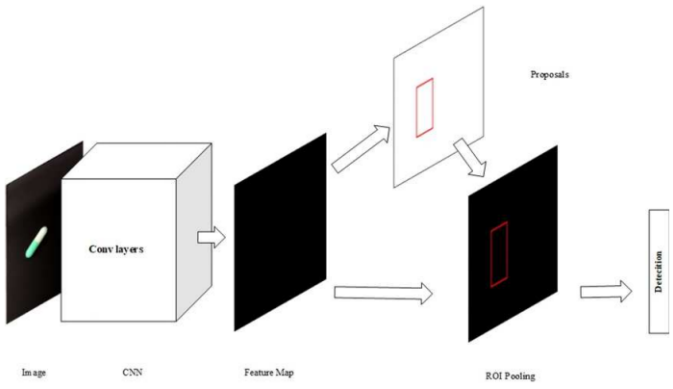


2 Object Detection Models

2.1 Model (1)

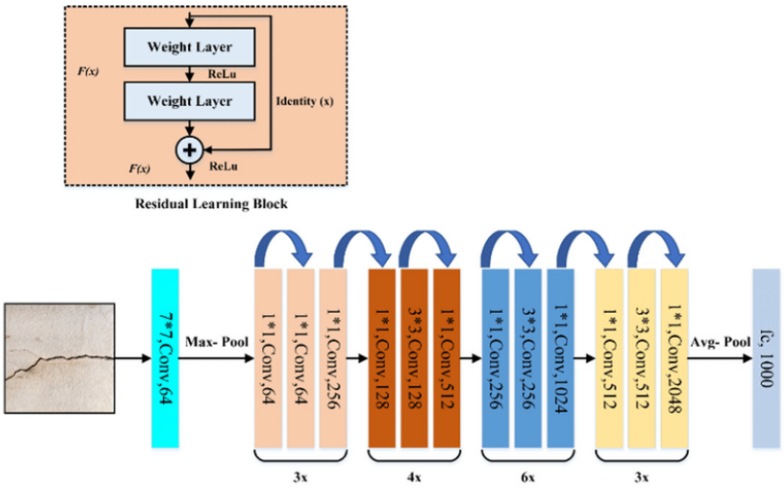
For the first model, we used Faster RCNN ResNet50 FPN. It's a Faster R-CNN model with a ResNet-50-FPN backbone.

Faster R-CNN is an object detection algorithm proposed by Ren et al. in 2015 consisting of four parts: feature extraction network, region proposal network, ROI Pooling, and a fully connected layer. Faster R-CNN is a modified version of R-CNN and Fast R-CNN algorithms. The difference between the two is that the Faster R-CNN algorithm avoids the computationally expensive selective search algorithm and uses the RPN to generate candidate regions instead.
This algorithm calculates the features of the whole image at once and thus does not involve repeated calculations, which greatly improves the detection speed of Faster R-CNN.

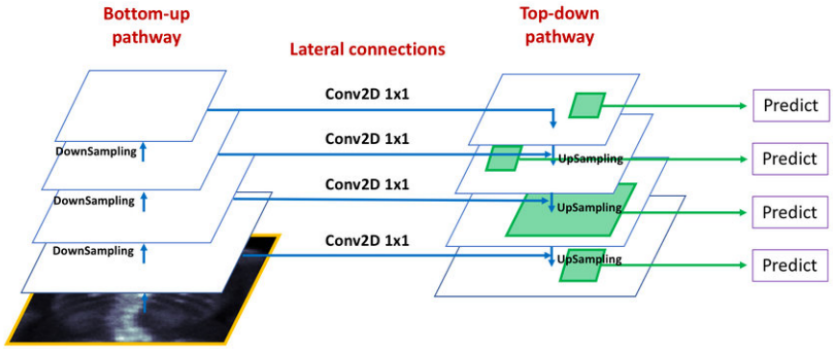


The 50-layer ResNet uses a bottleneck design for the building block. A bottleneck residual block uses 1×1 convolutions, known as a "bottleneck", which reduces the number of parameters and matrix multiplications. This enables much faster training of each layer. It uses a stack of three layers rather than two layers.

It consists of the following:
- A 7×7 kernel convolution alongside 64 other kernels with a 2-sized stride.
- A max pooling layer with a 2-sized stride.
- 9 more layers—3×3,64 kernel convolution, another with 1×1,64 kernels, and a third with 1×1,256 kernels. These 3 layers are repeated 3 times.
- 12 more layers with 1×1,128 kernels, 3×3,128 kernels, and 1×1,512 kernels, iterated 4 times.
- 18 more layers with 1×1,256 cores, and 2 cores 3×3,256 and 1×1,1024, iterated 6 times.
- 9 more layers with 1×1,512 cores, 3×3,512 cores, and 1×1,2048 cores iterated 3 times.
- Average pooling, followed by a fully connected layer with 1000 nodes, using the softmax activation function.



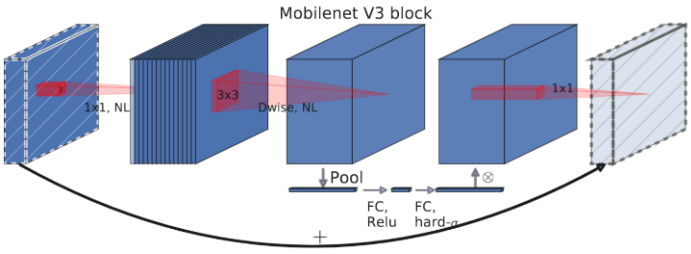
A Feature Pyramid Network, or FPN, is a feature extractor that takes a single-scale image of an arbitrary size as input, and outputs proportionally sized feature maps at multiple levels, in a fully convolutional fashion. This process is independent of the backbone convolutional architectures. It therefore acts as a generic solution for building feature pyramids inside deep convolutional networks to be used in tasks like object detection.



2.2 Model (2)

For the second model, we used FASTER RCNN MOBILENET V3 LARGE 320 FPN. It's a Low resolution Faster R-CNN model with a MobileNetV3-Large backbone tunned for mobile use cases.

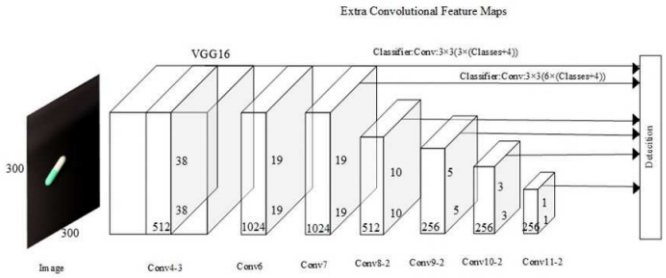
MobileNetV3 is a convolutional neural network that is tuned to mobile phone CPUs through a combination of hardware-aware network architecture search (NAS) complemented by the NetAdapt algorithm, and then subsequently improved through novel architecture advances. Advances include (1) complementary search techniques, (2) new efficient versions of nonlinearities practical for the mobile setting, (3) new efficient network design.



2.3 Model (3)

For the third model, we used SSD LITE320 MOBILENET V3 LARGE. It's a SSDlite model architecture with input size 320x320 and a MobileNetV3 Large backbone.

SSD was proposed by Wei Liu et al. and draws on the anchor mechanism of Faster R-CNN and the end-to-end one-step structure of the YOLO algorithm in which object classification and location regression are performed directly in the convolution stage. SSD is trained to obtain a set of fixed-sized bounding boxes and the class prediction scores of the targets in the bounding boxes.



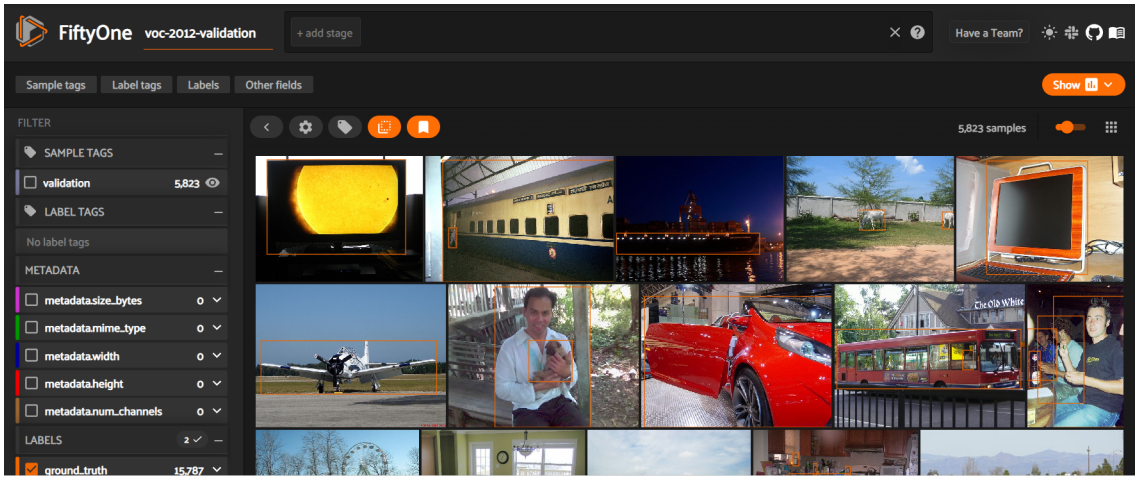
The SSDlite is an adaptation of SSD. SSDlite replaces the regular convolutions used on the original Heads with separable convolutions. Consequently, our implementation introduces new heads that use 3x3 Depthwise convolutions and 1x1 projections.

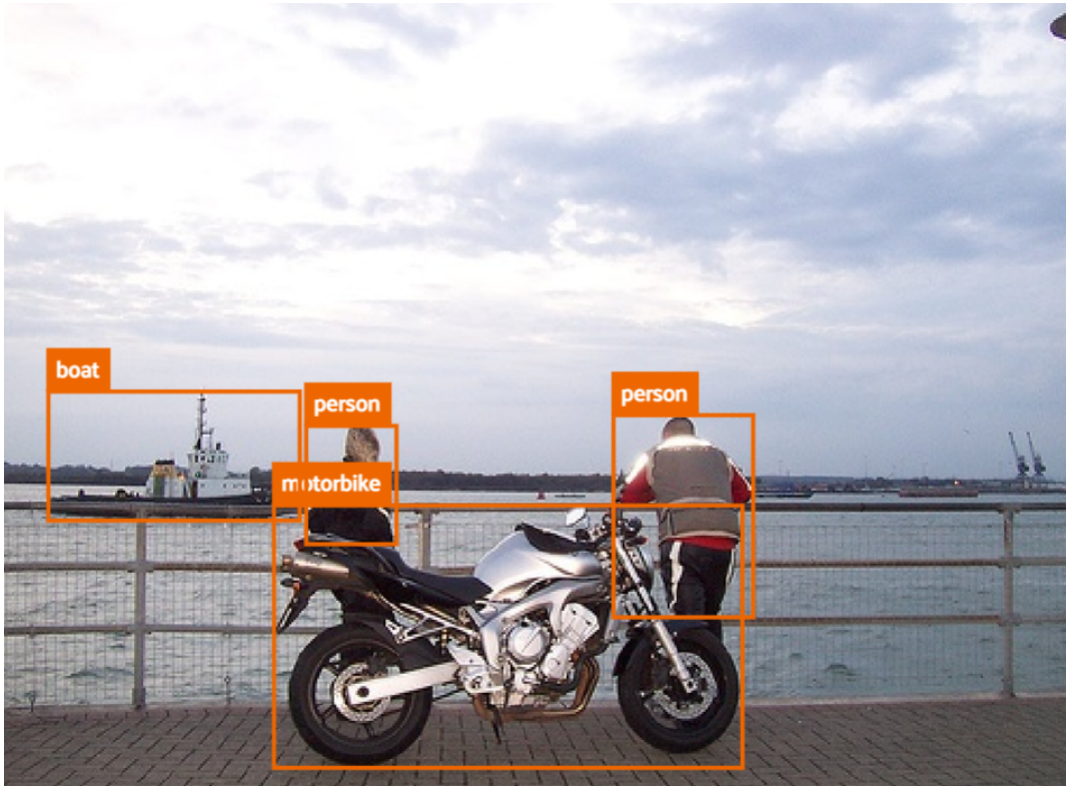
3 Scientific Background

3.1 VOC 2012 Dataset

We used VOC 2012 Dataset to test generalisation of our 3 pre-trained models, which is to test how well the models will perform when they're tested with data relatively different than the one they're trained on which is COCO 2017 Dataset.

The PASCAL Visual Object Classes (VOC) 2012 dataset contains 20 object categories including vehicles, household, animals, and other: aeroplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, TV/monitor, bird, cat, cow, dog, horse, sheep, and person. Each image in this dataset has pixel-level segmentation annotations, bounding box annotations, and object class annotations. This dataset has been widely used as a benchmark for object detection, semantic segmentation, and classification tasks.





3.2 Evaluation Metrics Used

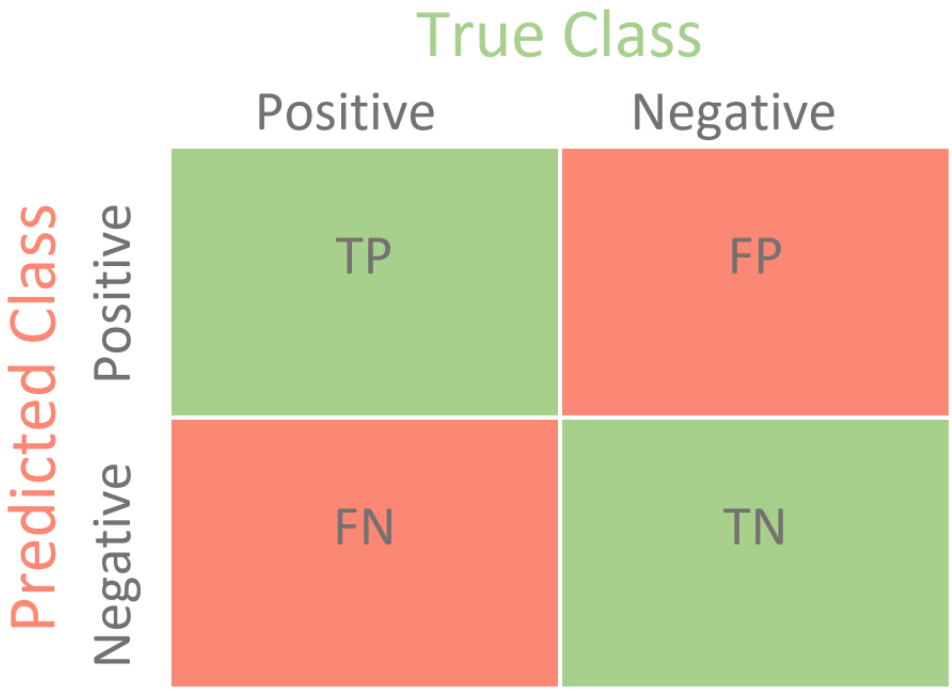
Confusion matrix:

- True Positives : The cases in which we predicted YES and the actual output was also YES.

- True Negatives : The cases in which we predicted NO and the actual output was NO.

- False Positives : The cases in which we predicted YES and the actual output was NO.

- False Negatives : The cases in which we predicted NO and the actual output was YES.



Confusion Matrix forms the basis for the other types of metrics.

Confidence level:

A confidence score is calculated as an evaluation standard. This confidence score shows the probability of the image being detected correctly by the algorithm and is given as a percentage. The scores are taken on the mean average precision at different IoU (Intersection over Union) thresholds.

The confidence score is given as the mean over all the precision scores for all thresholds.

$$\text{Confidence} = \text{Average precision} = \frac{1}{|\text{Thresholds}|} \sum_T \frac{\text{TP}}{\text{TP+FP+FN}}$$

Precision:

It is the number of correct positive results divided by the number of positive

results predicted by the classifier.

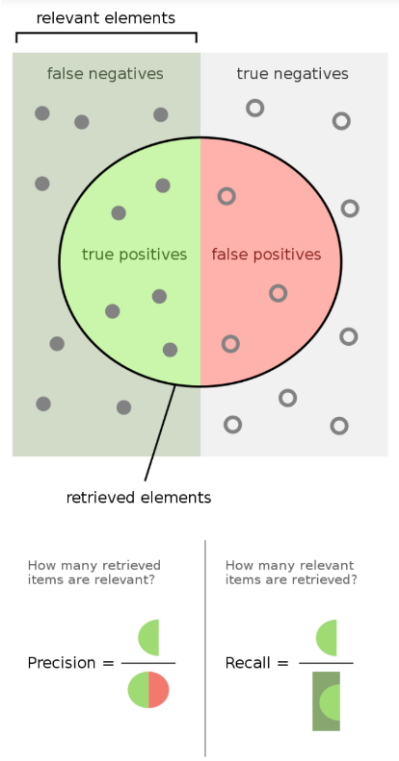
$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

Precision

Recall:

It is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive).

$$\boldsymbol{Recall = \frac{TruePositive}{TruePositive + FalseNegative}}$$



Support:

Support may be defined as the number of samples of the true response that lies in each class of target values.

F1-score:

F1 Score is the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1]. It tells you how precise your classifier is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances).

High precision but lower recall, gives you an extremely accurate, but it then misses a large number of instances that are difficult to classify. The greater the F1 Score, the better is the performance of our model.

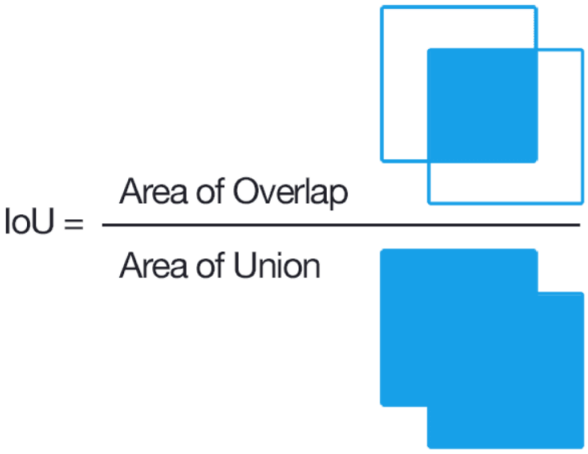
$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

F1 Score

F1 Score tries to find the balance between precision and recall.

Intersection over Union (IoU):

Intersection over Union is an evaluation metric used to measure the accuracy of an object detector on a particular dataset.



Average Precision (AP):

Average Precision is calculated as the weighted mean of precisions at each threshold; the weight is the increase in recall from the prior threshold.

Here is a summary of the steps to calculate the AP:

- Generate the prediction scores using the model.
- Convert the prediction scores to class labels.
- Calculate the confusion matrix—TP, FP, TN, FN.
- Calculate the precision and recall metrics.
- Calculate the area under the precision-recall curve.
- Measure the average precision.

Mean Average Precision (mAP):

Mean Average Precision(mAP) is a metric used to evaluate object detection models and its values are calculated over recall values from 0 to 1.

mAP formula is based on the following sub metrics:

- Confusion Matrix
- Intersection over Union(IoU)
- Recall
- Precision

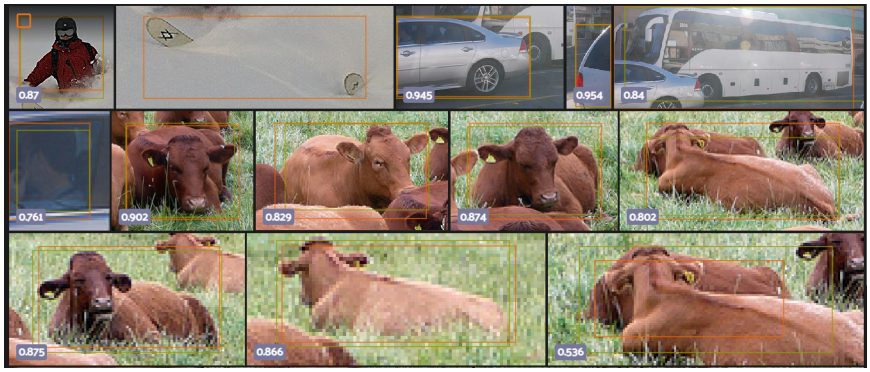
Mean Average Precision is the average of AP of each class. However, the interpretation of AP and mAP varies in different contexts.

4 Evaluate Models

4.1 Model (1)

4.1.1 COCO Dataset 2017

Here, we'll evaluate the results obtained when testing the model on COCO Dataset 2017, validation split.
IoU:



Mean Average Precision (mAP):

mAP of this model

```
[ ]  print(results.mAP())

     0.3693043447294809
```

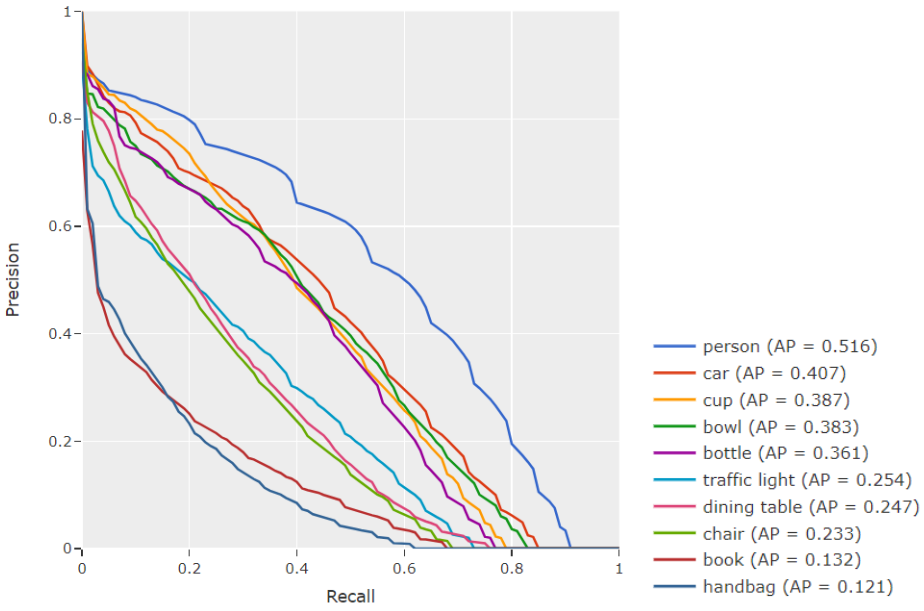
True positives, false positives and false negatives count:

```
View stages:
    1. Take(size=100, seed=51)
    2. ToEvaluationPatches(eval_key='eval', config=None)
{'fp': 2867, 'fn': 133, 'tp': 630}
```

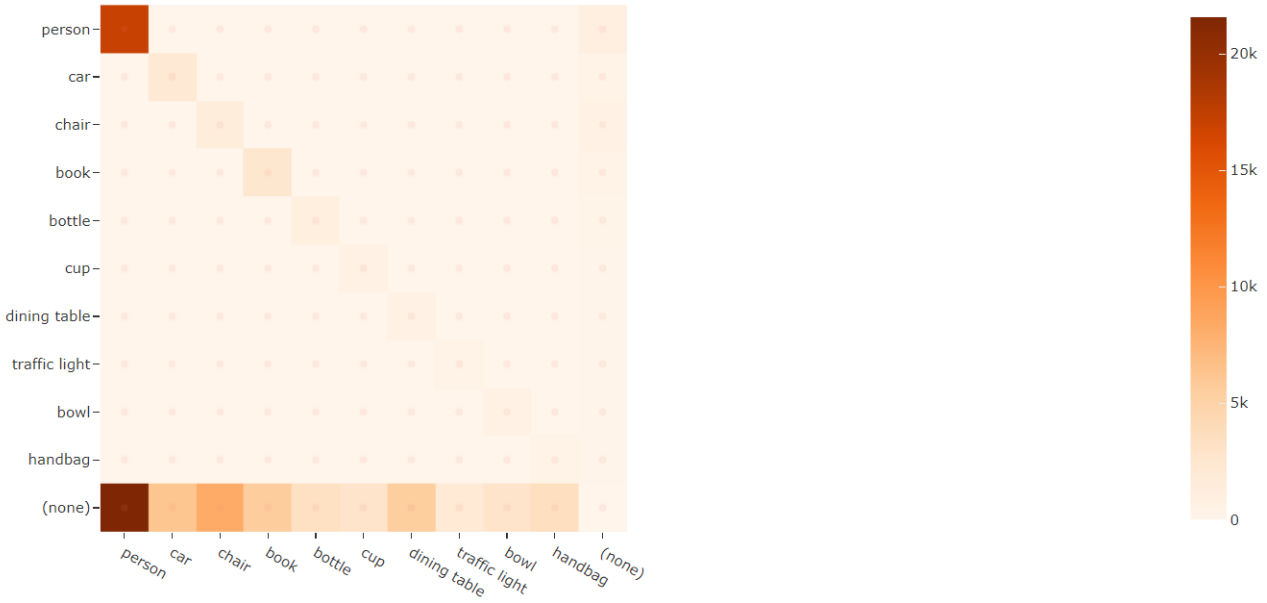
Top 10 most common classes classification report:

	precision	recall	f1-score	support
person	0.44	0.94	0.60	18206
car	0.25	0.87	0.39	2414
chair	0.15	0.73	0.26	2074
book	0.31	0.87	0.46	2947
bottle	0.25	0.82	0.38	1354
cup	0.21	0.80	0.33	960
dining table	0.09	0.76	0.16	703
traffic light	0.19	0.73	0.30	654
bowl	0.16	0.83	0.27	652
handbag	0.09	0.61	0.15	540
micro avg	0.30	0.89	0.45	30504
macro avg	0.21	0.80	0.33	30504
weighted avg	0.35	0.89	0.50	30504

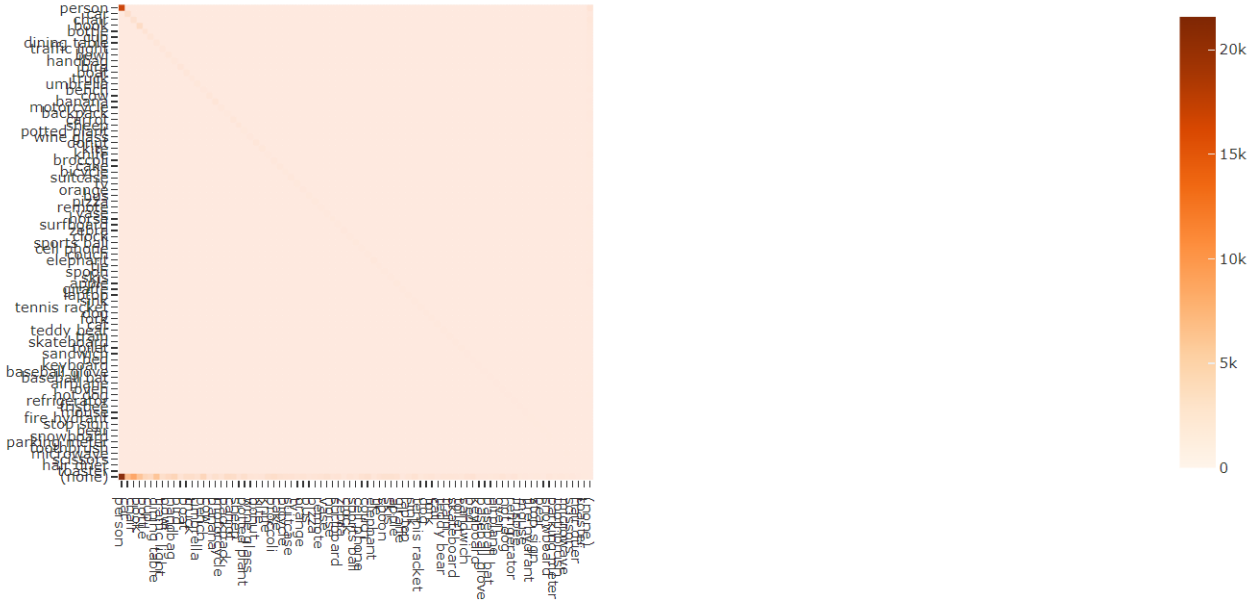
Precision vs. Recall for top 10 classes:



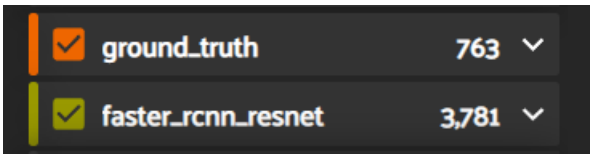
Confusion matrix for top 10 classes:



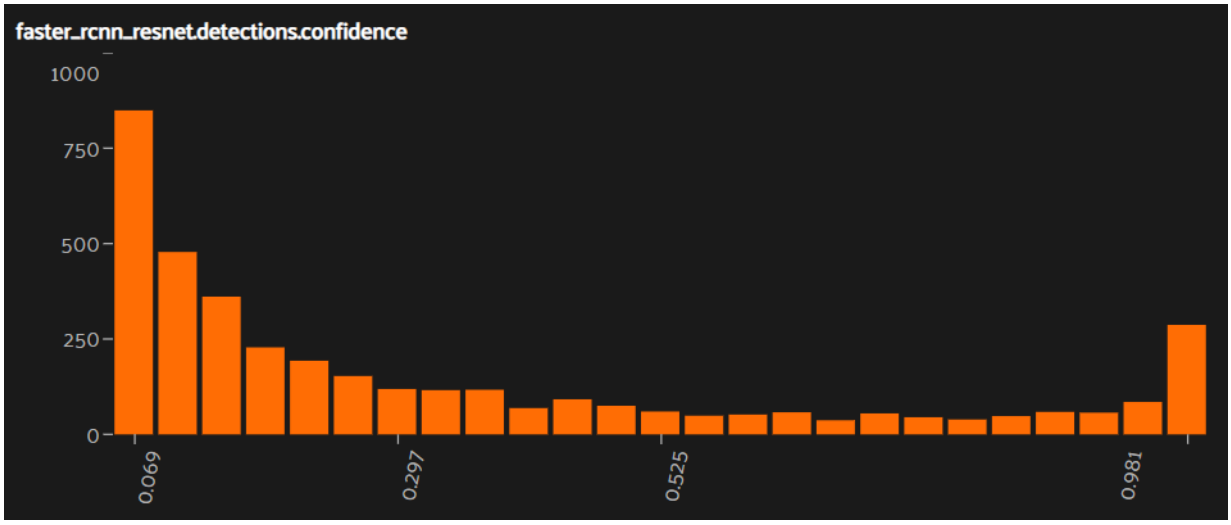
Confusion matrix for all 80 classes:



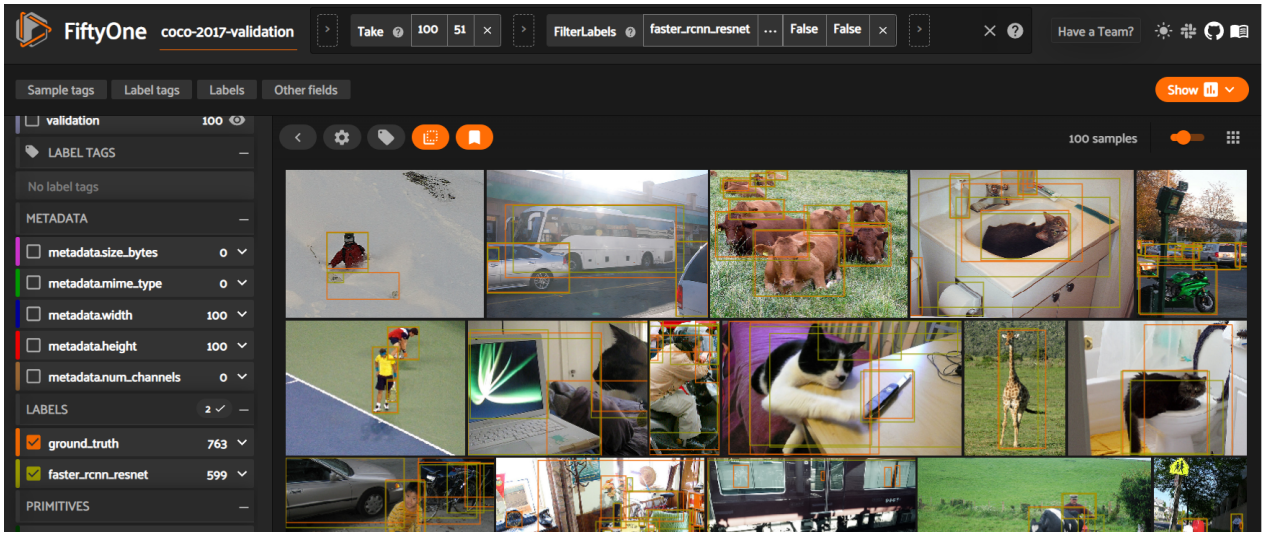
Number of actual objects in 100 samples vs number of objects detected by model:

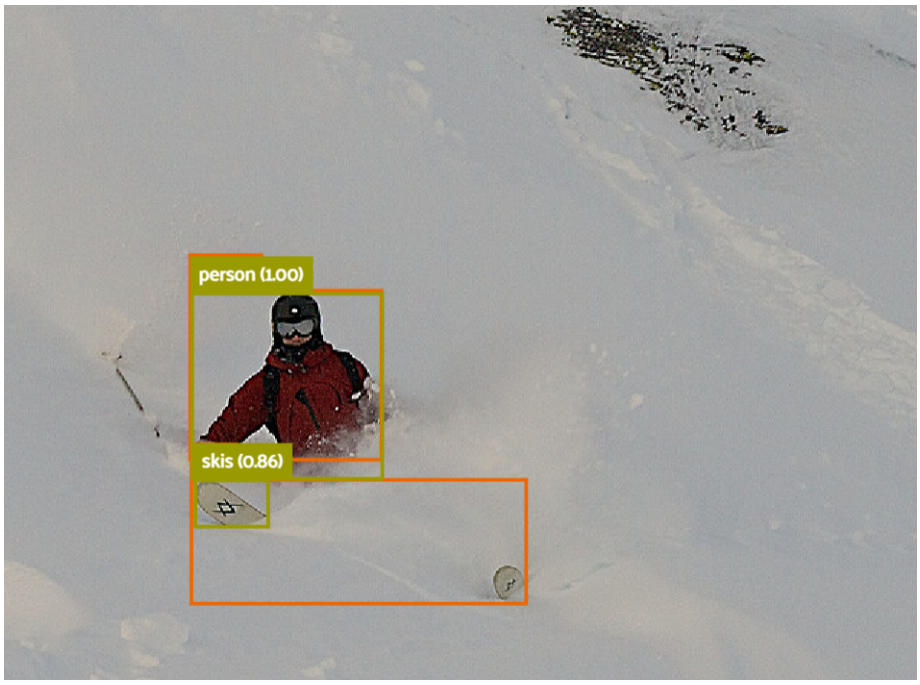


Confidence level:

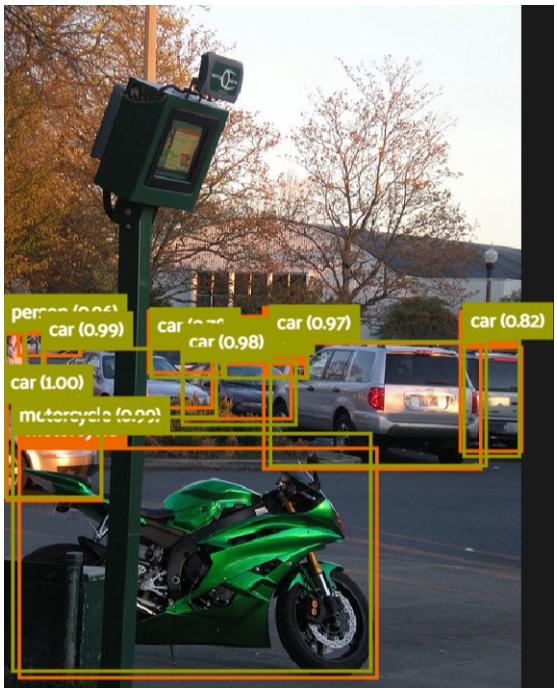


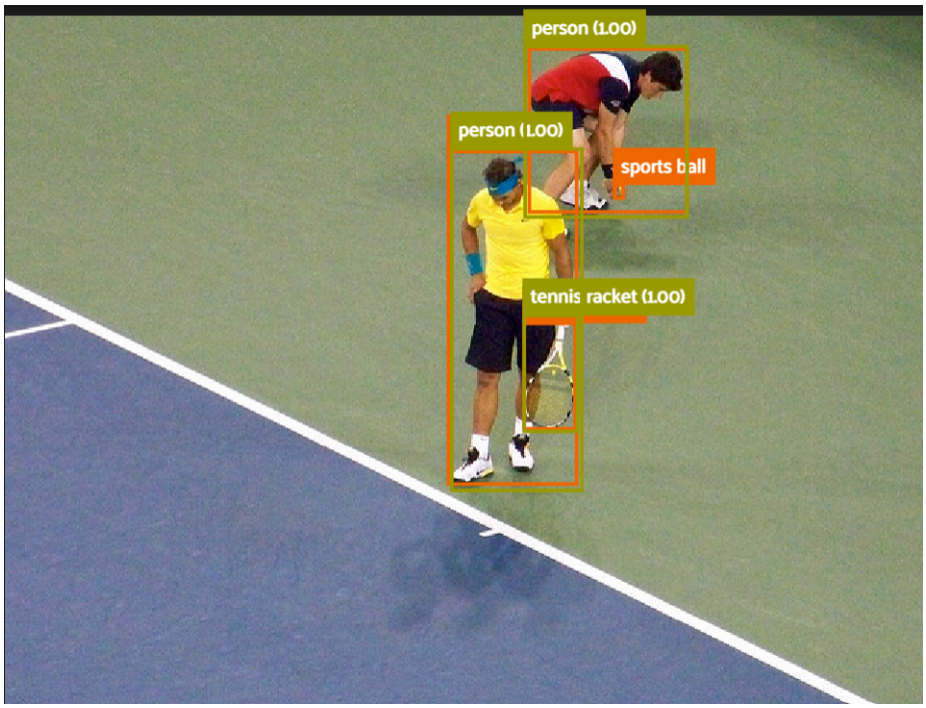
Success cases:

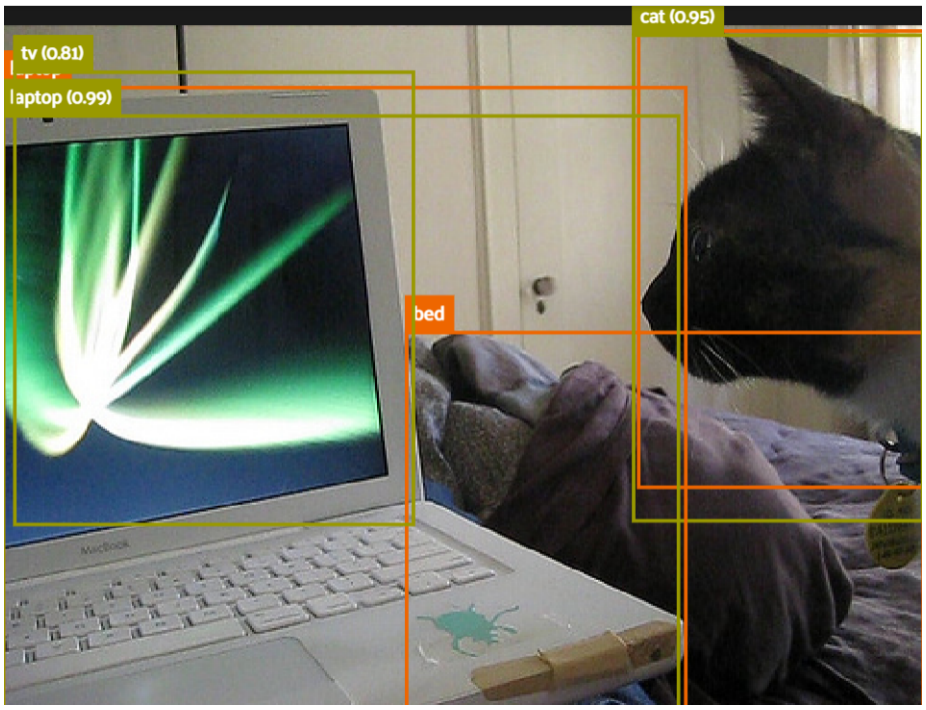



person (1.00)
skis (0.86)


bus (0.98)
car (1.00)
person (0.91)
car (0.97)


car (0.99)
car (0.98)
car (0.97)
car (0.82)
car (1.00)


person (1.00)
person (1.00)
sports ball
tennis racket (1.00)



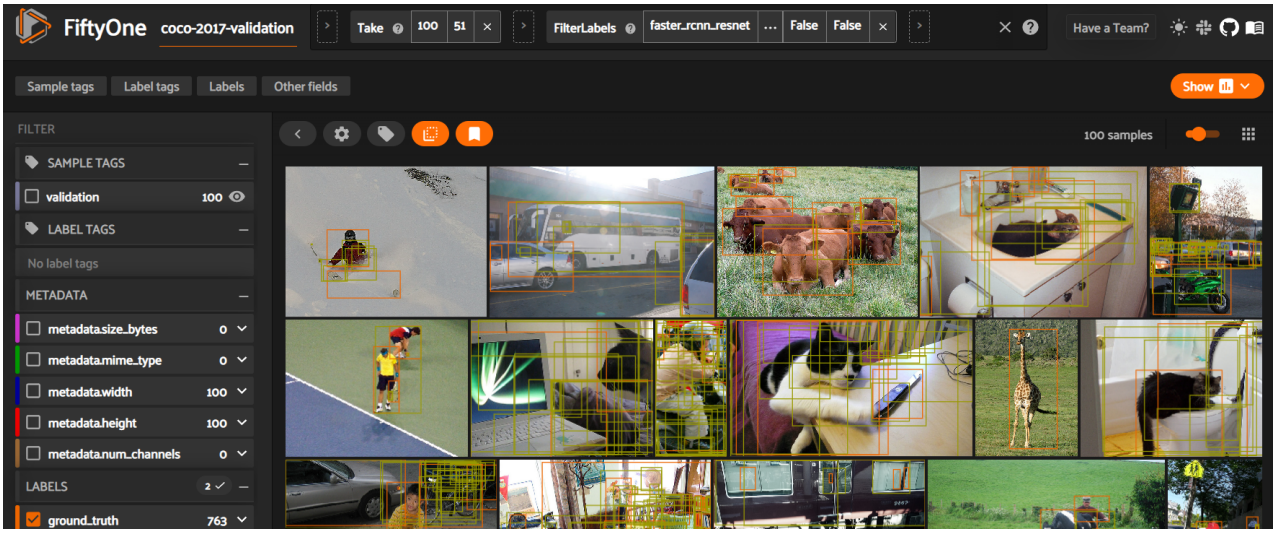
Comments:

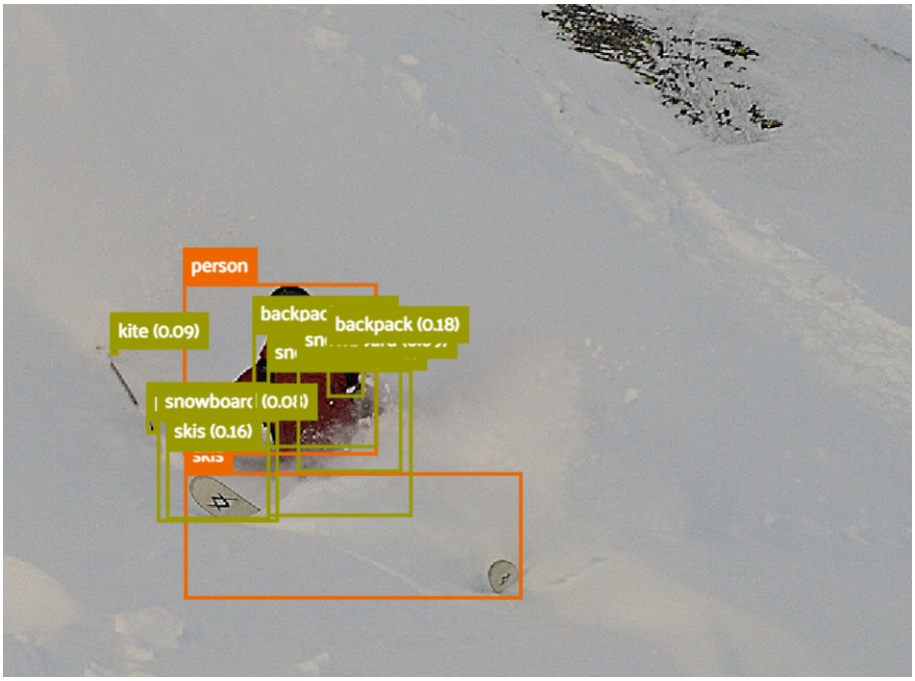
- All these results have confidence of 0.75 or more.

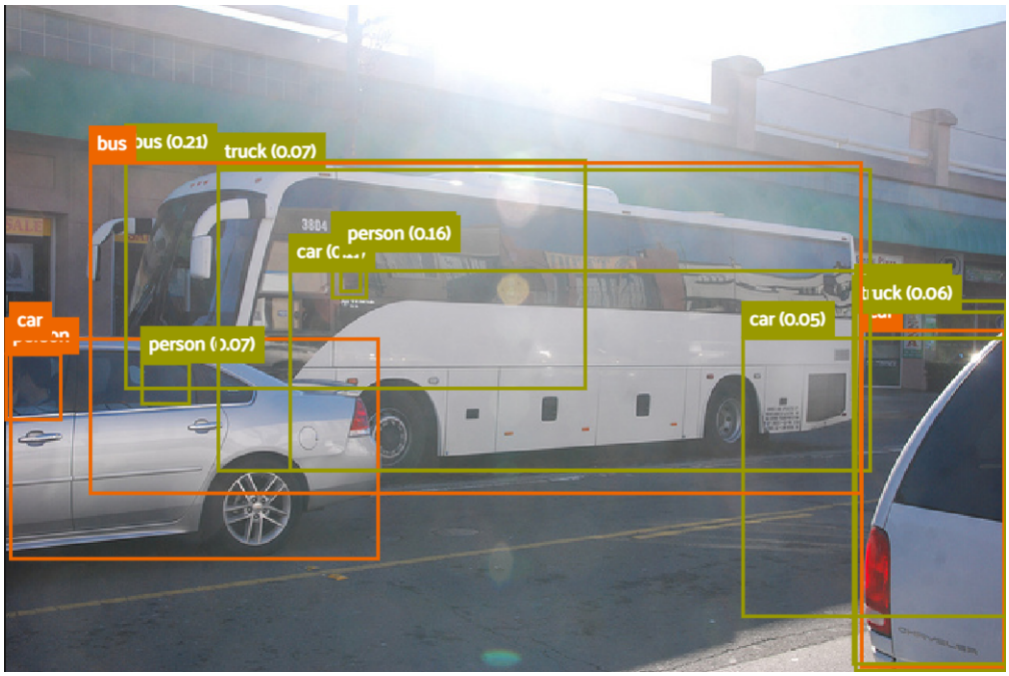
- This model can detect objects even if part of object is hidden.

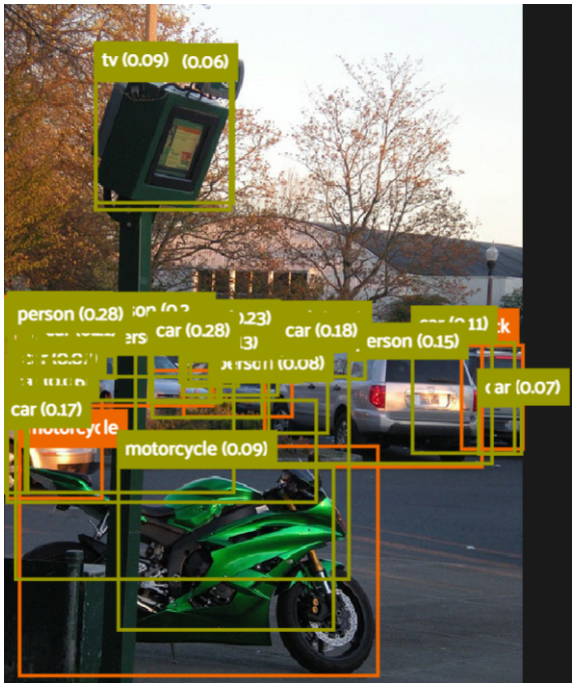
- The model can predict more objects than the ones represented by the ground truth boundary boxes with good accuracies which means it's well trained.

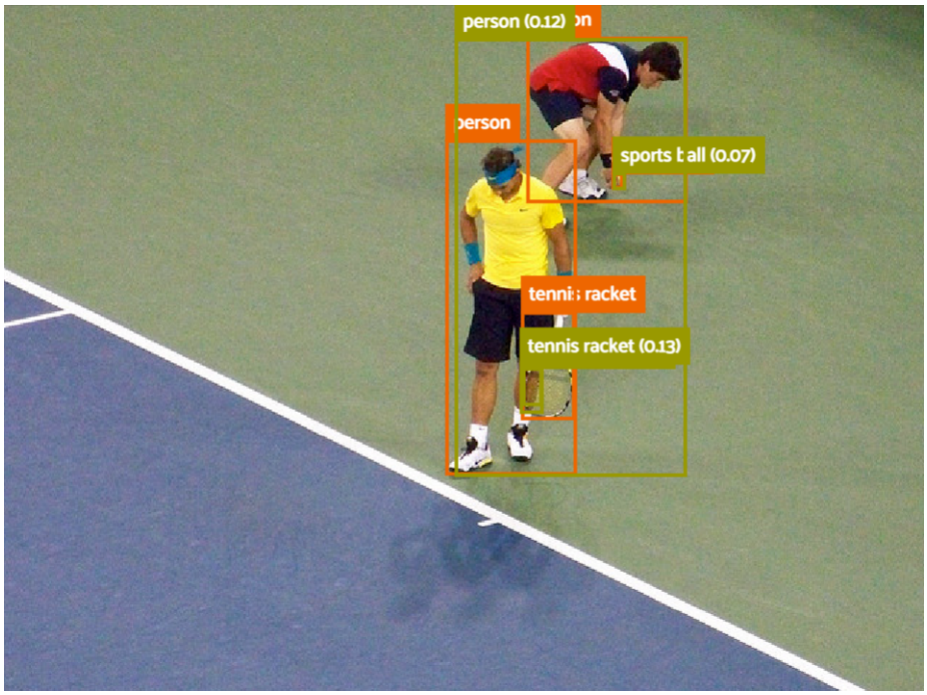
Failure cases:

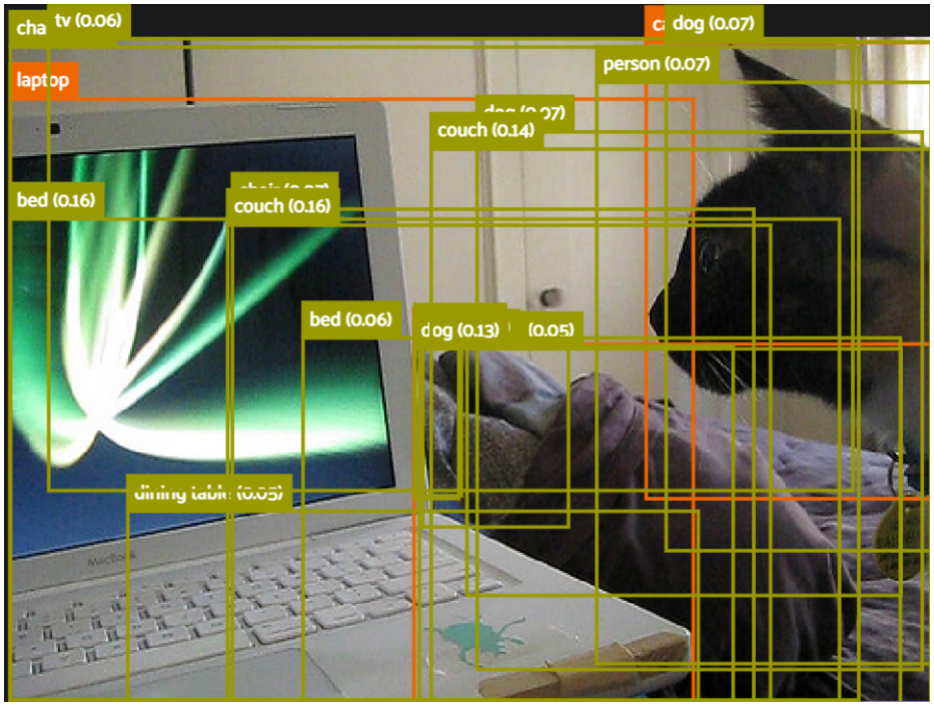





bus
truck (0.07)
person (0.16)
car
car (0.05)


tv (0.09)
(0.06)
person (0.28)
car (0.28)
car (0.18)
car (0.17)
motorcycle (0.09)


person (0.12)
sports ball (0.07)
tennis racket
tennis racket (0.13)


tv (0.06)
dog (0.07)
laptop
person (0.07)
couch (0.14)
bed (0.16)
couch (0.16)
bed (0.06)
dog (0.13)
(0.05)

Comments:

- All these results have confidence of 0.3 or less.

- When parts of the objects are hidden, the same object can be detected more than once causing the existence of many redundant bounding boxes.

- The model sometimes detects 2 objects of the same class together as one object if there is overlapping between them.

- The model doesn't seem to be able to differentiate between different animals when their colour or shape is relatively similar.

4.1.2 VOC 2012 Dataset

Here, we'll evaluate the results obtained when testing the model on VOC 2012 Dataset, validation split.
IoU:

Mean Average Precision (mAP):

mAP of this model on VOC 2012

```
[ ]  print(results_voc2012.mAP())

     0.3711368196554111
```

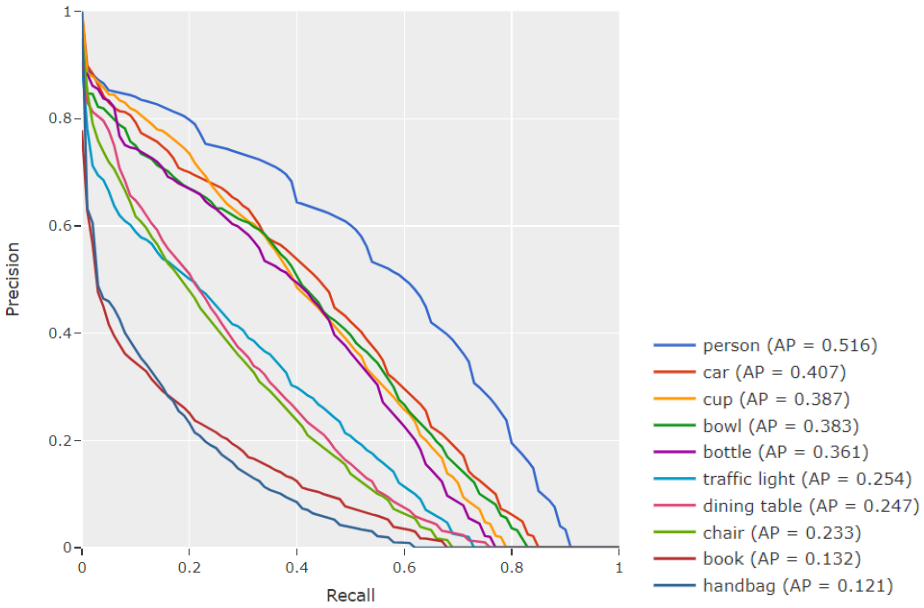
True positives, false positives and false negatives count:

```
View stages:
    1. Take(size=100, seed=51)
    2. ToEvaluationPatches(eval_key='eval', config=None)
{'tp': 208, 'fn': 31, 'fp': 1856}
```

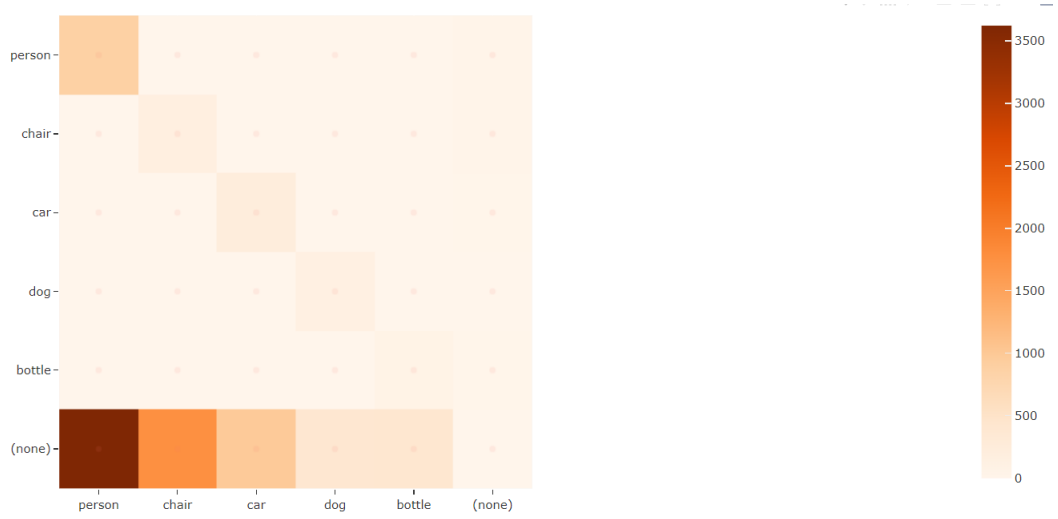
Top 5 most common classes classification report:

	precision	recall	f1-score	support
person	0.20	0.96	0.33	921
chair	0.09	0.85	0.17	209
car	0.19	0.96	0.32	237
dog	0.25	0.98	0.40	139
bottle	0.14	0.90	0.25	80
micro avg	0.17	0.94	0.29	1586
macro avg	0.17	0.93	0.29	1586
weighted avg	0.18	0.94	0.31	1586

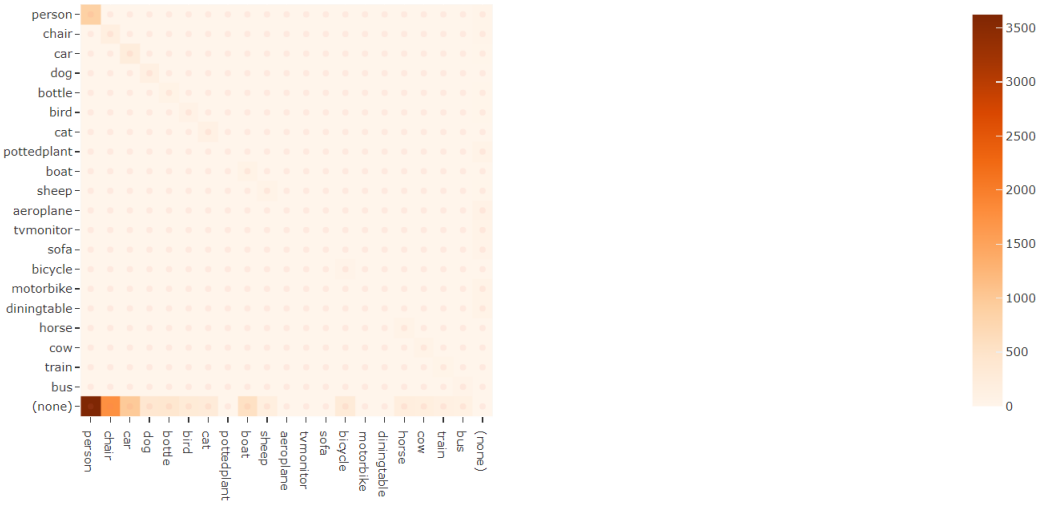
Precision vs. Recall for top 5 classes:



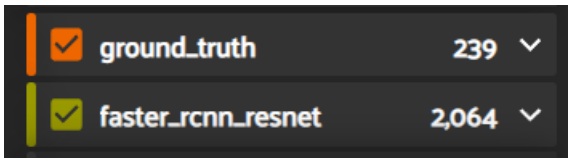
Confusion matrix for top 5 classes:



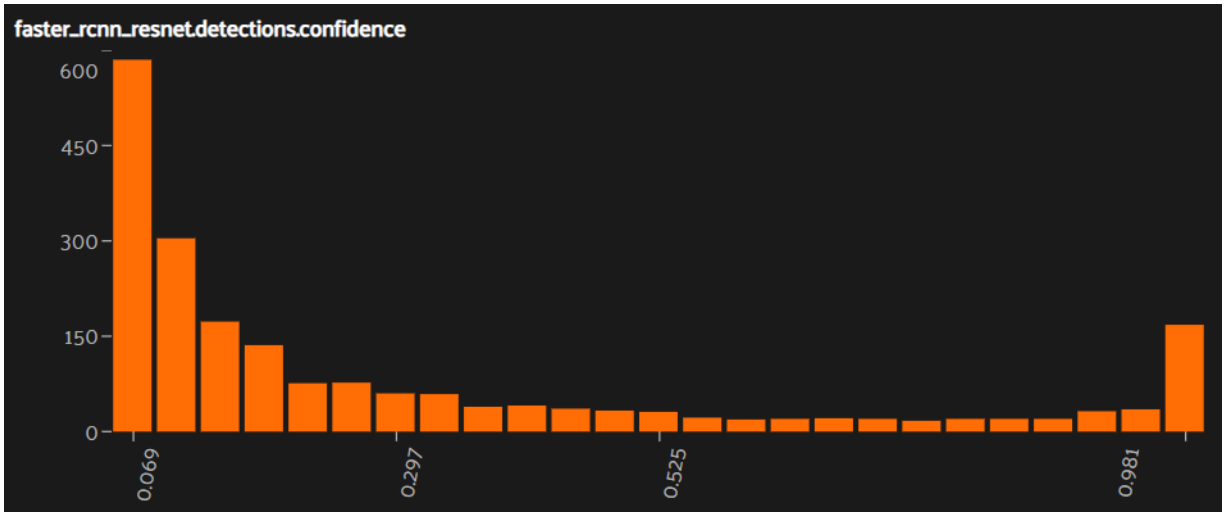
Confusion matrix for all 20 classes:



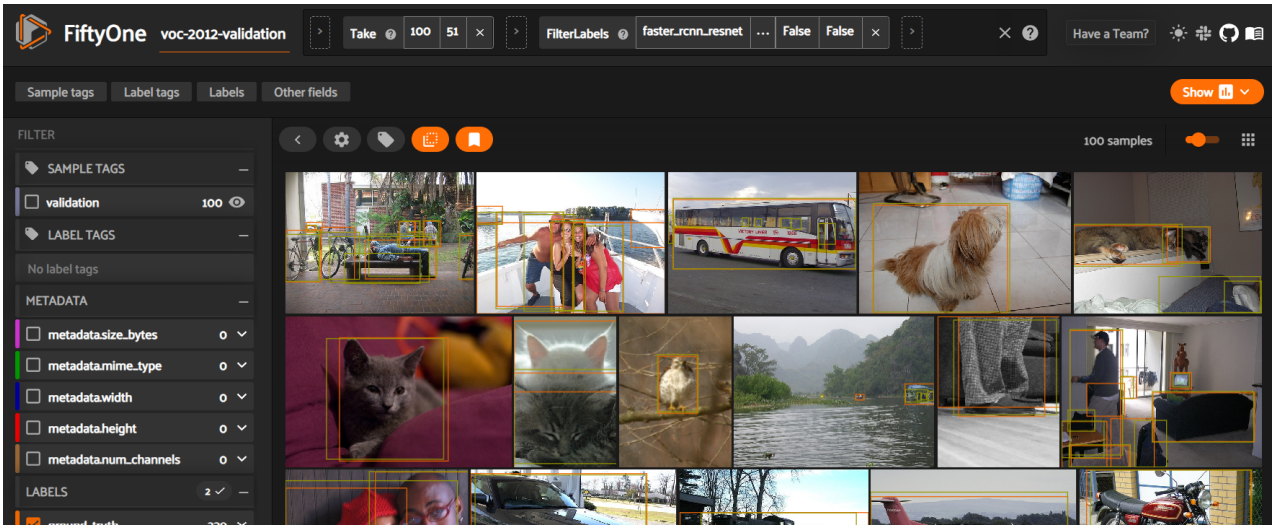
Number of actual objects in 100 samples vs number of objects detected by model:

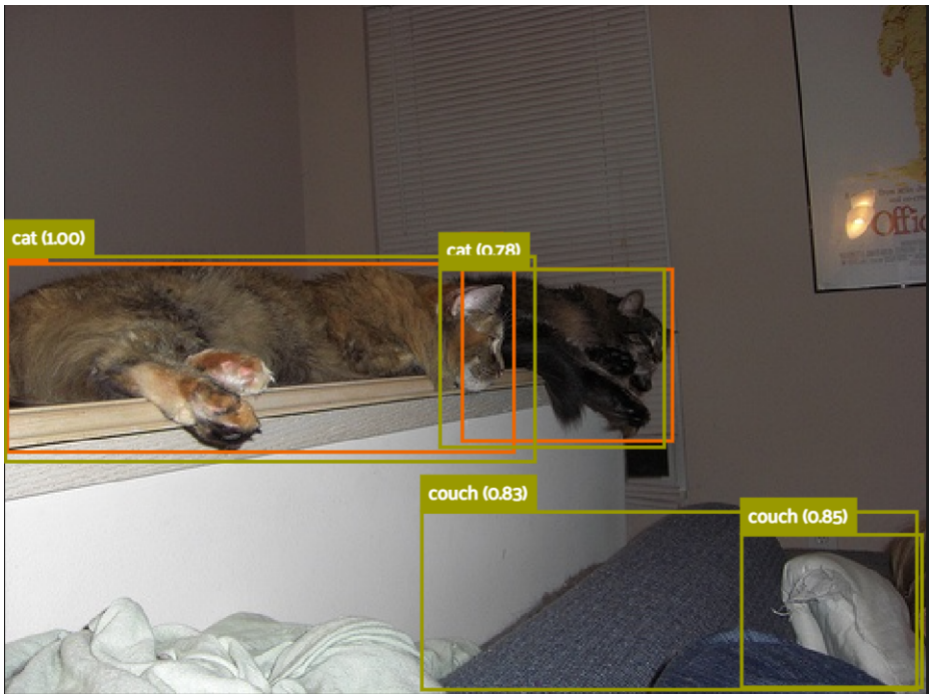


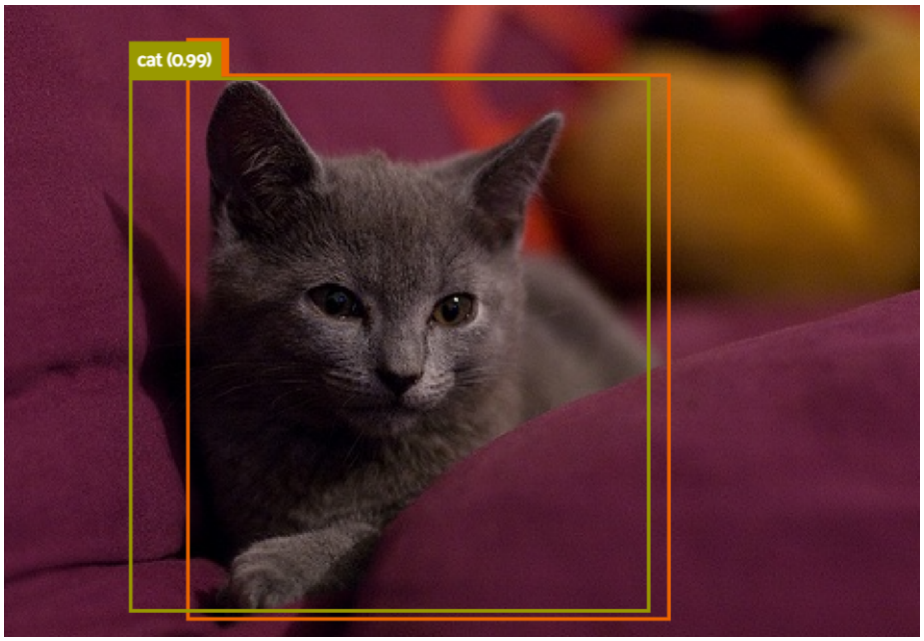
Confidence level:

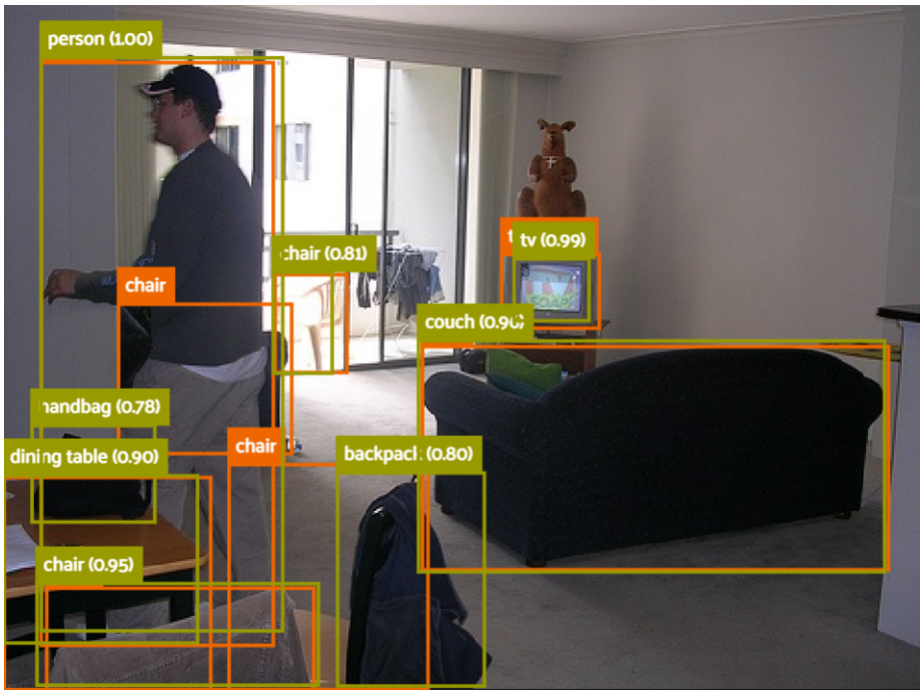


Success cases:




cat (1.00)
cat (0.78)
couch (0.83)
couch (0.85)


cat (0.99)


person (1.00)
chair (0.81)
tv (0.99)
chair
couch (0.90)
handbag (0.78)
chair
dining table (0.90)
backpack (0.80)
chair (0.95)


boat
boat (1.00)
kite (0.86)
person (0.99)
person (0.97)
person (0.99)
boat (0.93)
EXPLORER

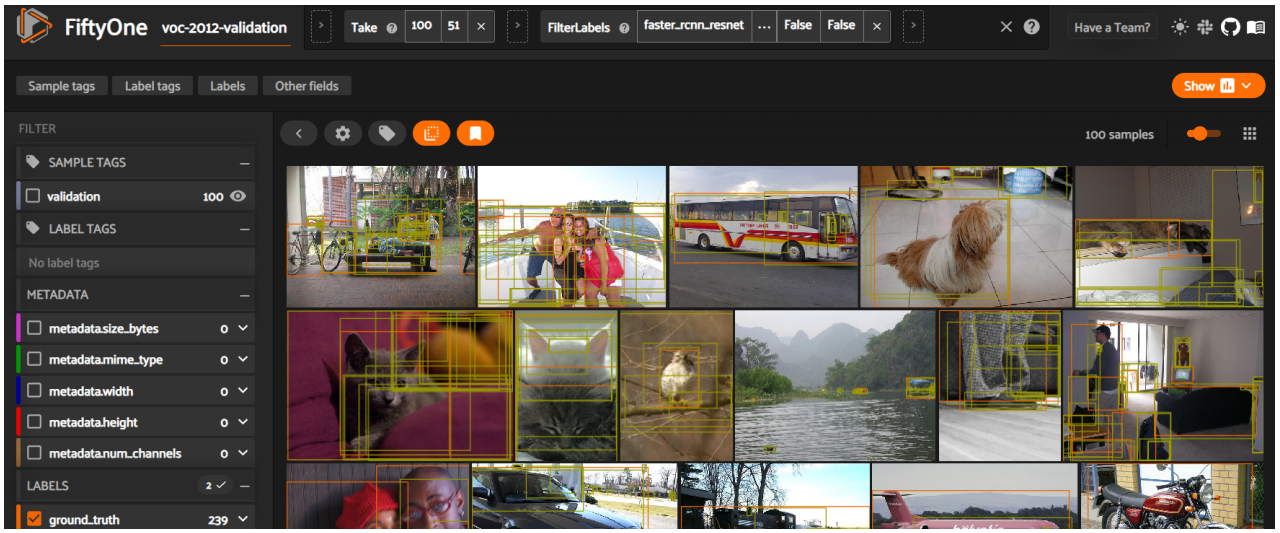
Comments:

- All these results have confidence of 0.75 or more.

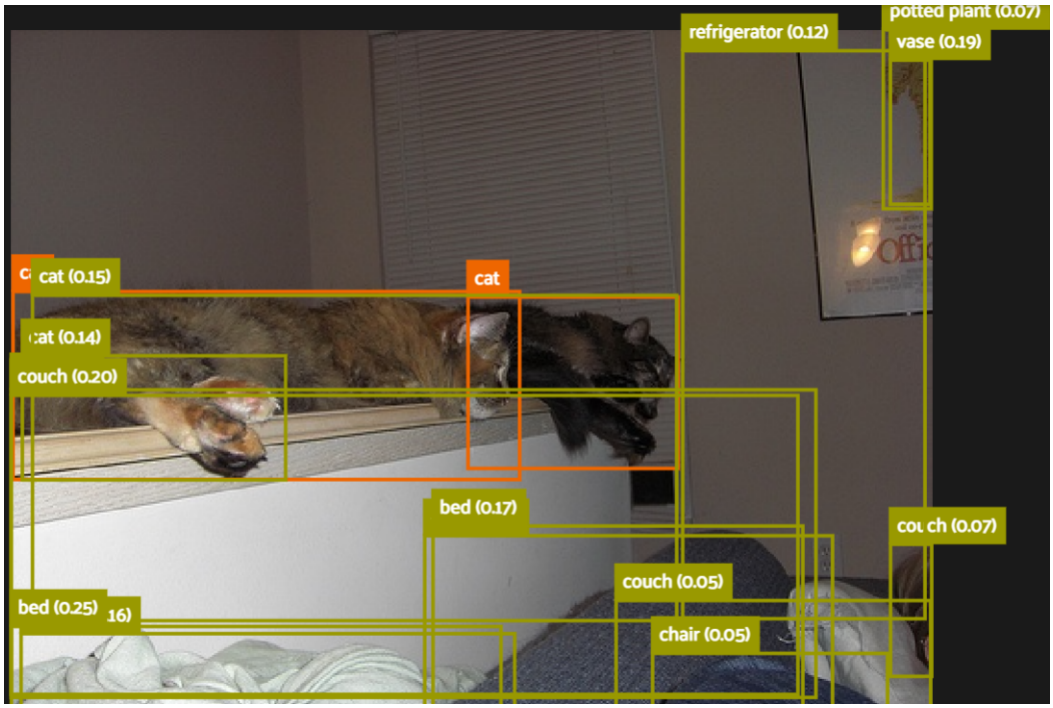
- Due to the limited number of classes in this dataset, it's easier and more accurate to detect people even when they're far at a distant because there less other classes to confuse them with.

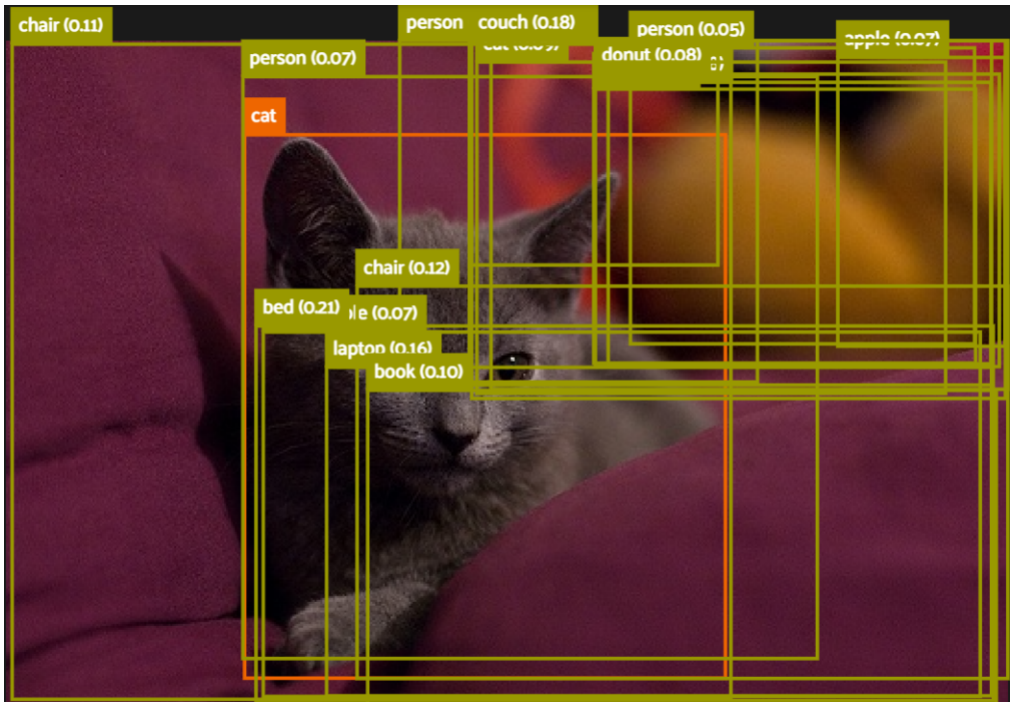
- It can recognise different shapes of the same objects like chairs, armchairs, etc...

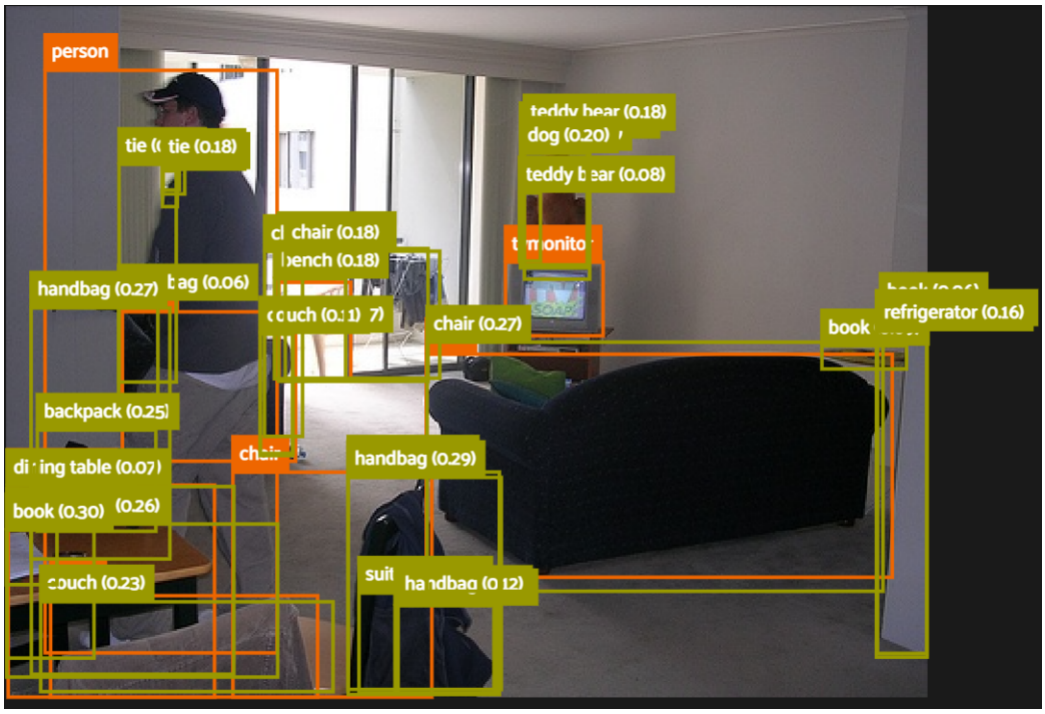
Failure cases:

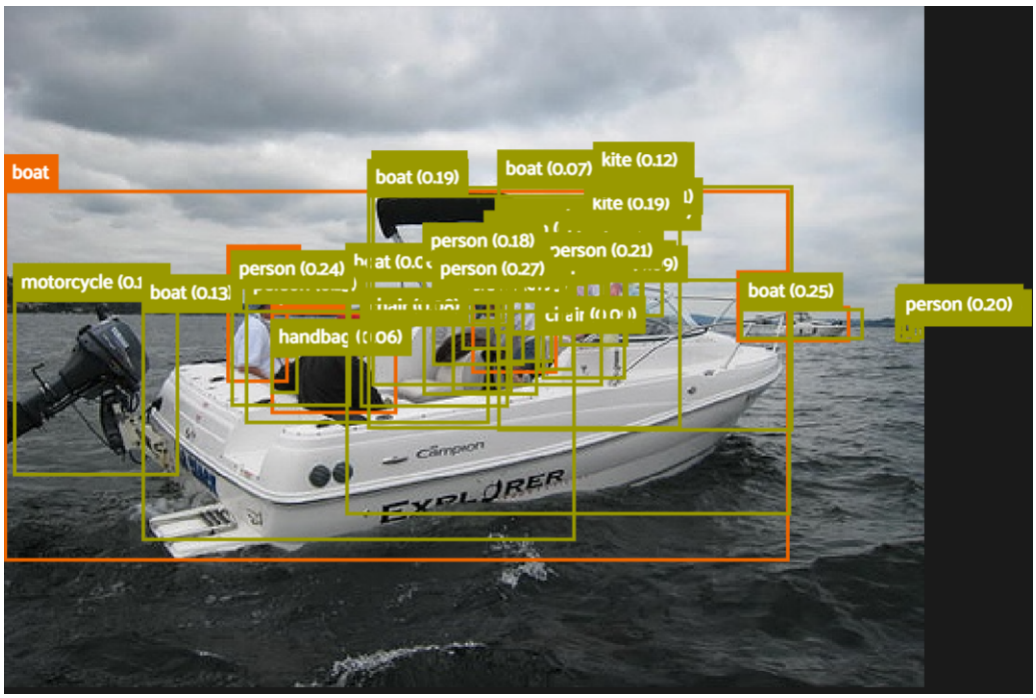



bus (0.25)
bus (0.23)
bus (0.11)
bus (0.14)
person (0.17)
VICTORY LINER
1888


refrigerator (0.12)
vase (0.19)
cat (0.15)
cat
cat (0.14)
couch (0.20)
bed (0.17)
couch (0.05)
chair (0.05)
bed (0.25)


chair (0.11)
person
couch (0.18)
person (0.05)
person (0.07)
donut (0.08)
cat
chair (0.12)
bed (0.21)
book (0.10)


person
tie (0.18)
teddy bear (0.18)
dog (0.20)
teddy bear (0.08)
chair (0.18)
bench (0.18)
handbag (0.27)
bag (0.06)
chair (0.27)
refrigerator (0.16)
backpack (0.25)
handbag (0.29)
book (0.30)
(0.26)
couch (0.23)
handbag (0.12)



Comments:

- All these results have confidence of 0.3 or less.

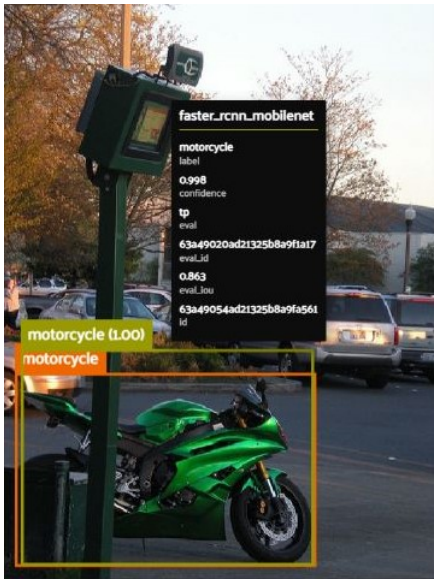
- A drawback to the limited number of classes in this dataset is that the model is trained on more classes so it tries to map many detections to few limited classes which sometimes give bad results.

- It mistakes parts of objects for other objects.

4.2 Model (2)

4.2.1 COCO Dataset 2017

Here, we'll evaluate the results obtained when testing the model on COCO Dataset 2017, validation split.
IoU:



Mean Average Precision (mAP):

mAP of this model

```
[ ]  print(results.mAP())
```

```
0.22744898884151493
```

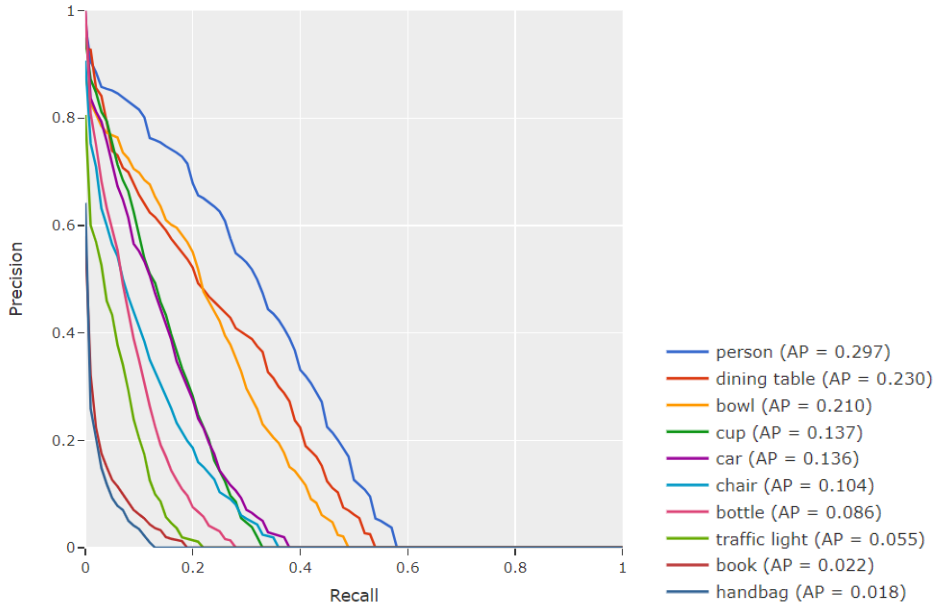
True positives, false positives and false negatives count:

```
View stages:
    1. Take(size=100, seed=51)
    2. ToEvaluationPatches(eval_key='eval', config=None)
{'tp': 380, 'fn': 383, 'fp': 1080}
```

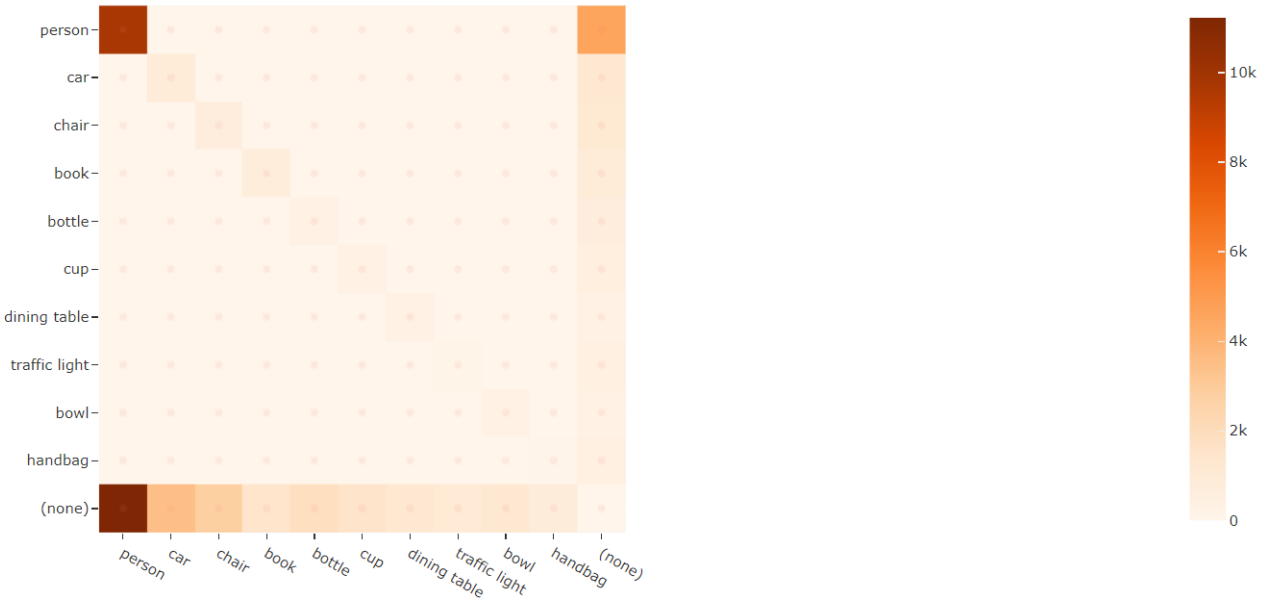
Top 10 most common classes classification report:

	precision	recall	f1-score	support
person	0.47	0.68	0.55	14397
car	0.21	0.44	0.28	2120
chair	0.21	0.39	0.27	1893
book	0.35	0.46	0.40	1711
bottle	0.17	0.34	0.23	1116
cup	0.17	0.35	0.23	919
dining table	0.23	0.53	0.32	698
traffic light	0.11	0.21	0.15	637
bowl	0.21	0.50	0.29	644
handbag	0.08	0.13	0.10	540
micro avg	0.34	0.56	0.42	24675
macro avg	0.22	0.40	0.28	24675
weighted avg	0.36	0.56	0.44	24675

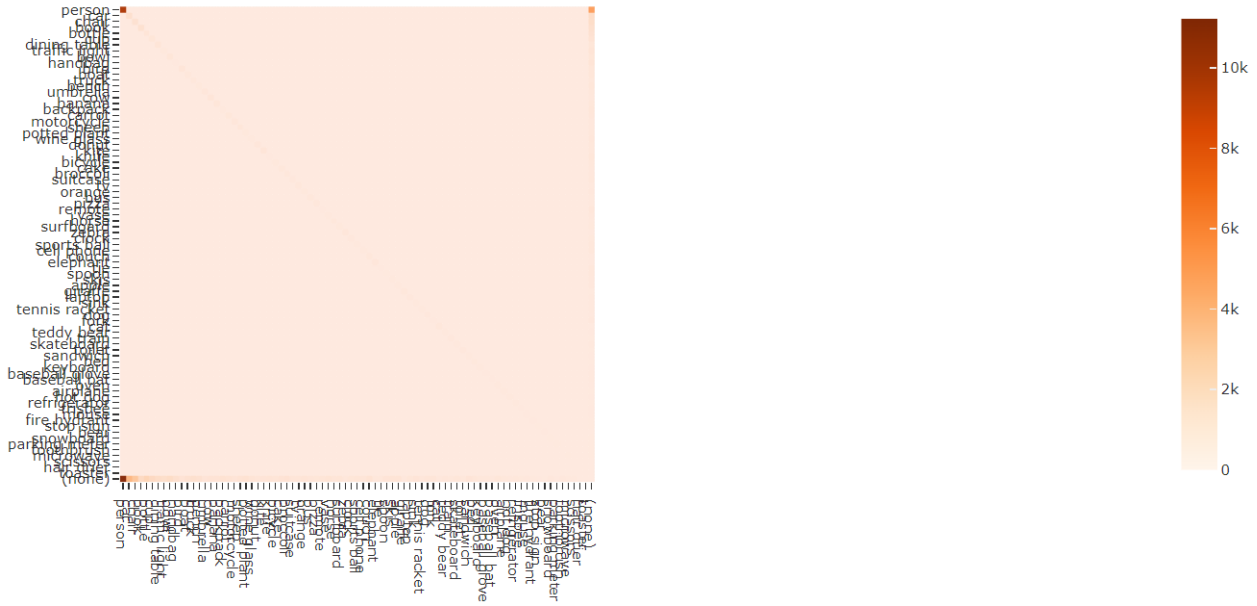
Precision vs. Recall for top 10 classes:



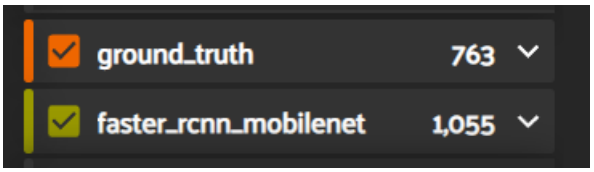
Confusion matrix for top 10 classes:



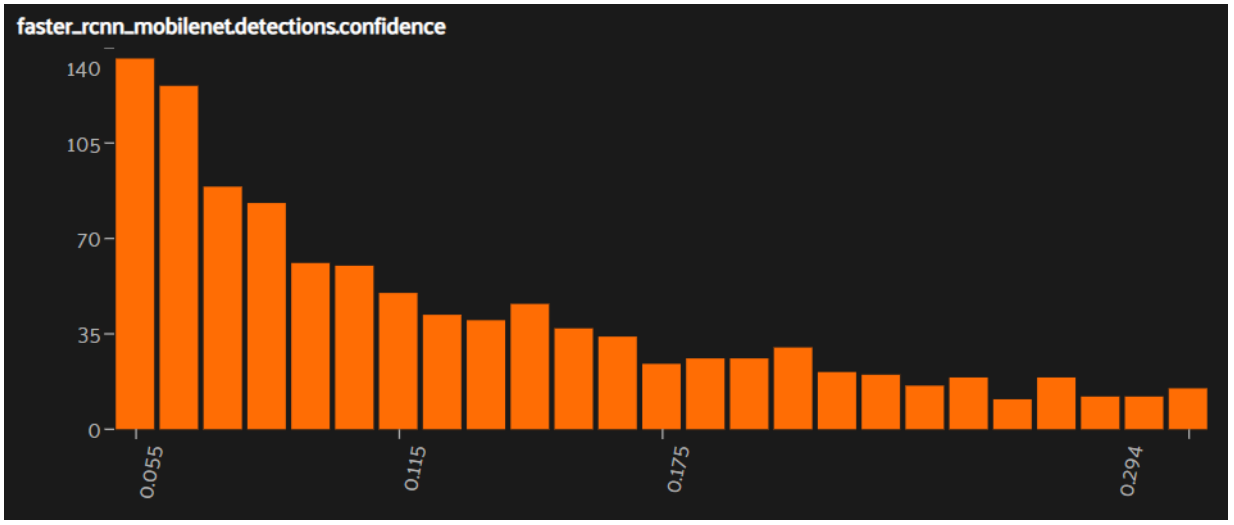
Confusion matrix for all 80 classes:



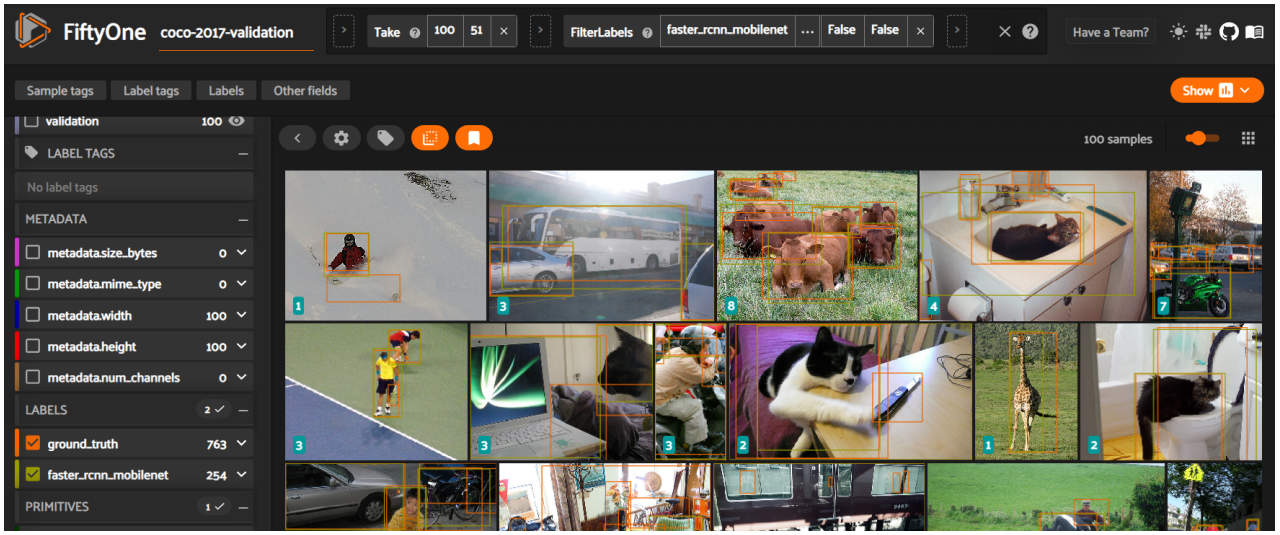
Number of actual objects in 100 samples vs number of objects detected by model:

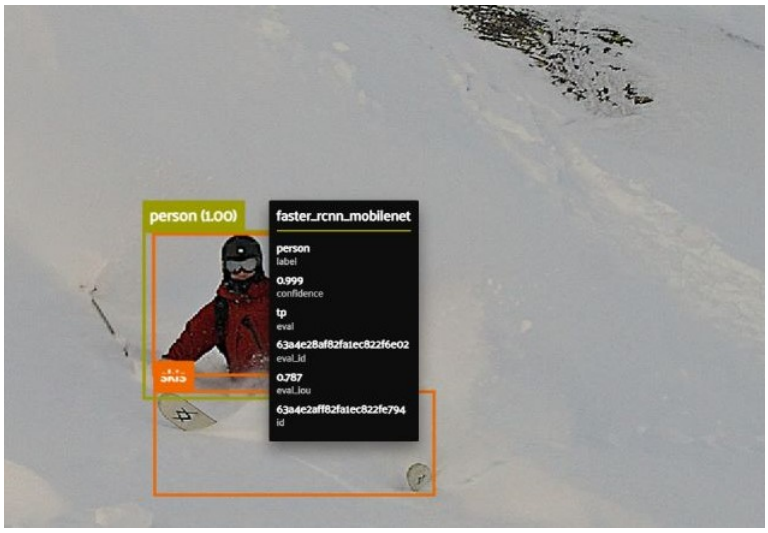


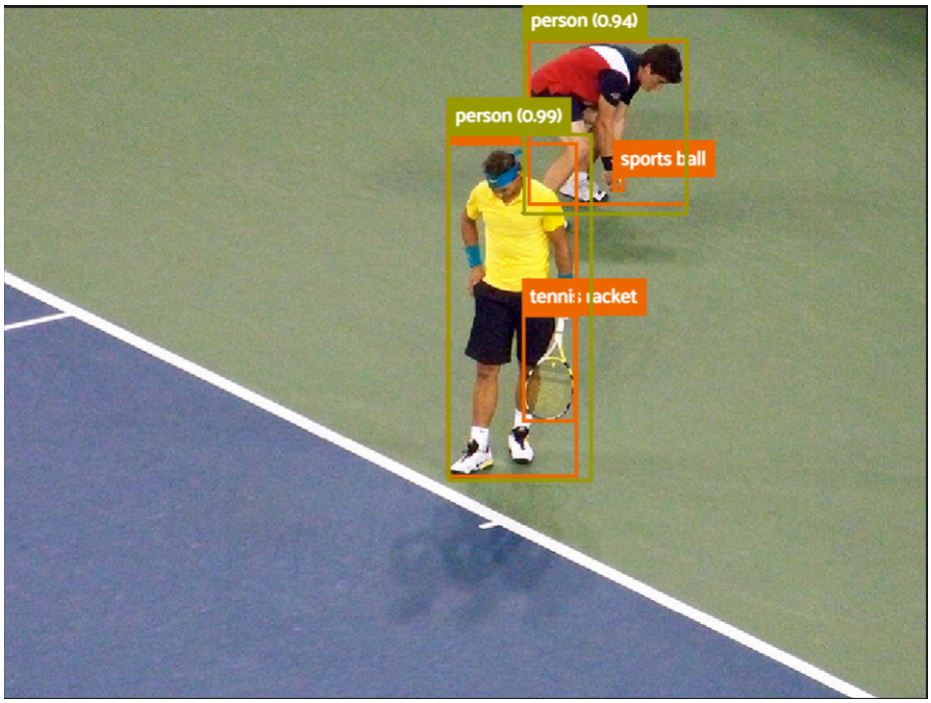
Confidence level:

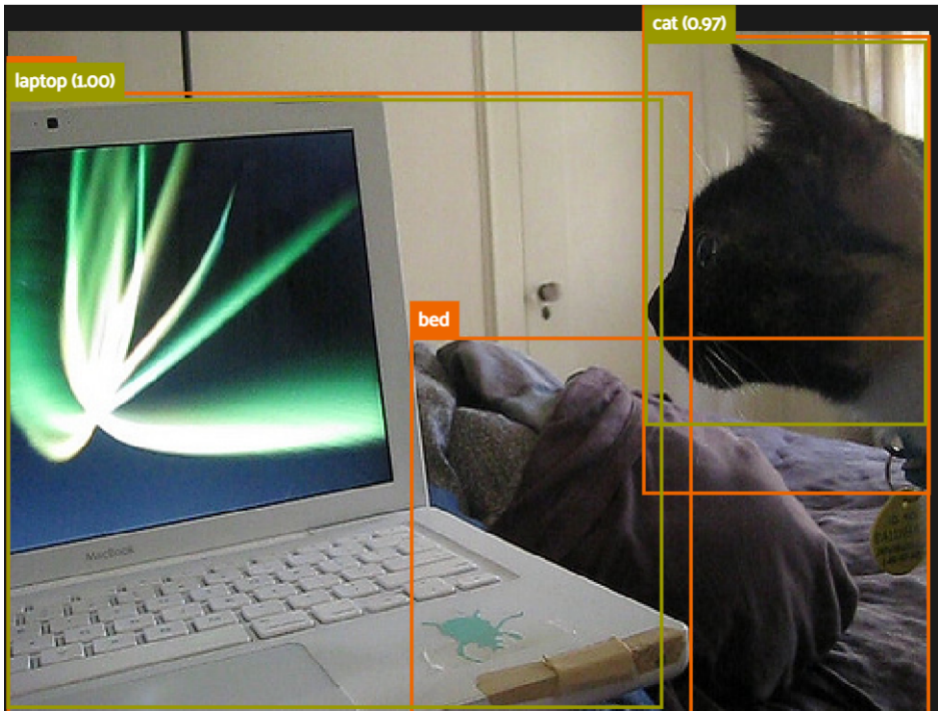


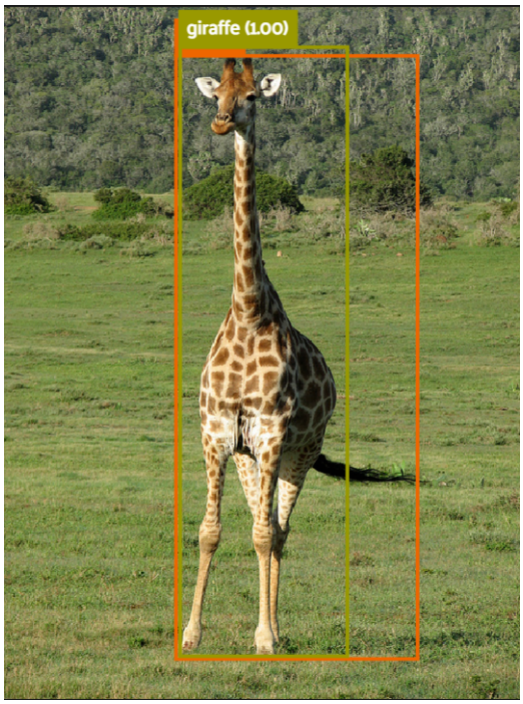
Success cases:




person (1.00)
faster_rcnn_mobilenet
person
label
0.999
confidence
tp
eval
63a4e28af82fa1ec822f6e02
eval_id
0.787
eval_iou
63a4e2aff82fa1ec822fe794
id


person (0.94)
person (0.99)
sports ball
tennis racket


cat (0.97)
laptop (1.00)
bed


giraffe (1.00)

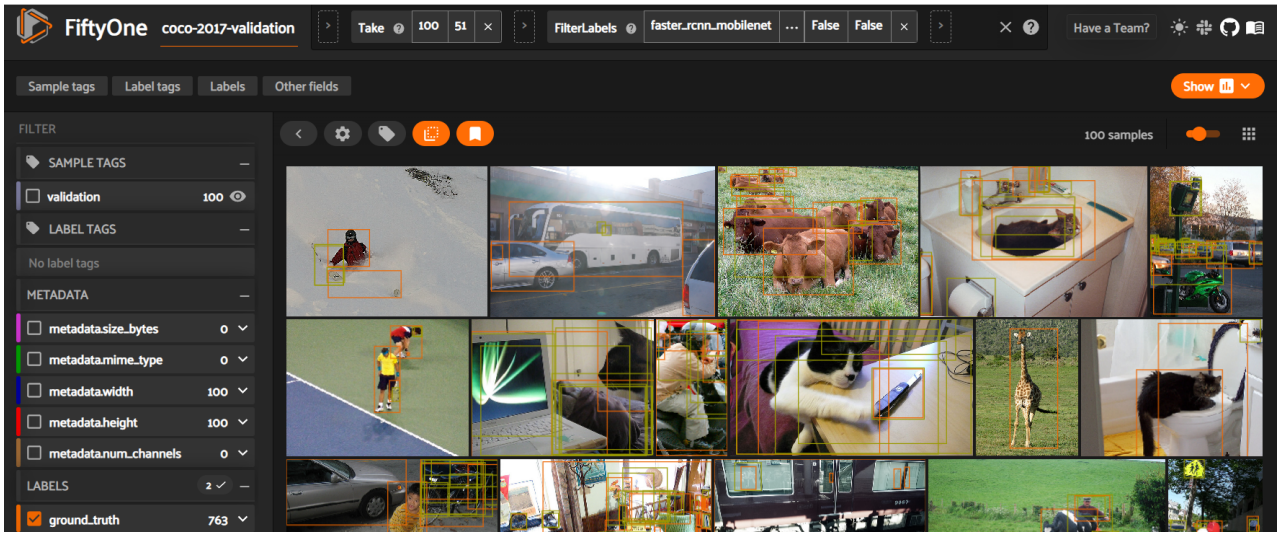
Comments:

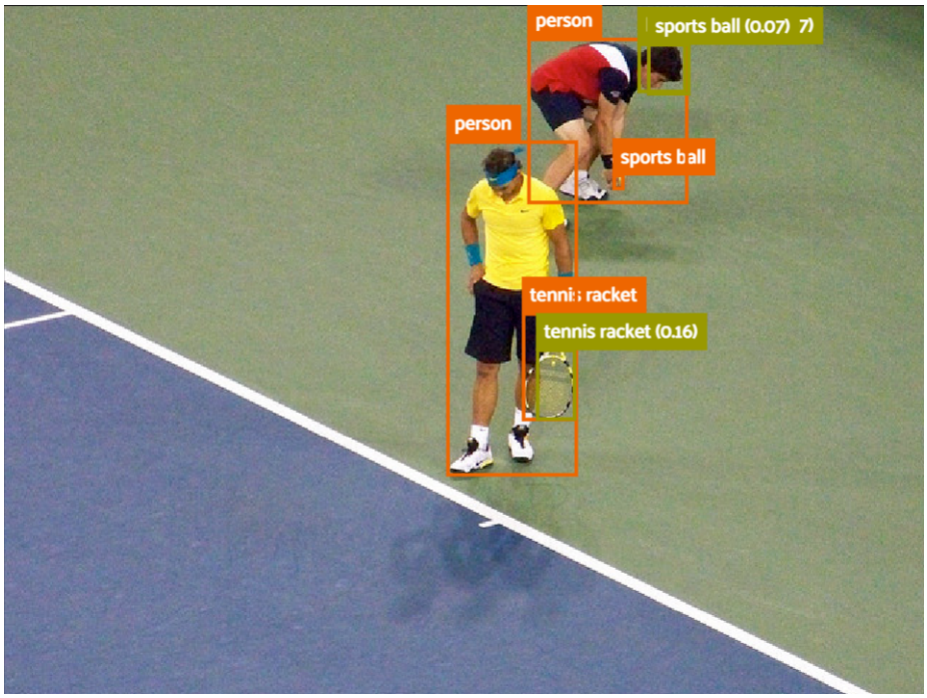
- All these results have confidence of 0.75 or more.

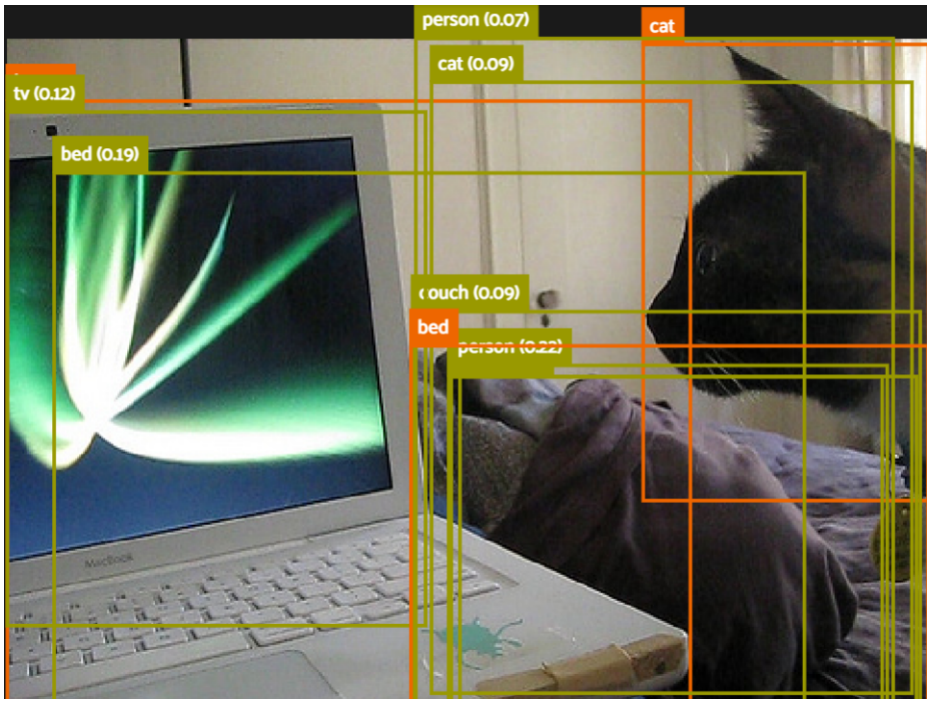
- We can notice that the model is really good at detecting people which explains why they're the most detected class given our top 10 classes.

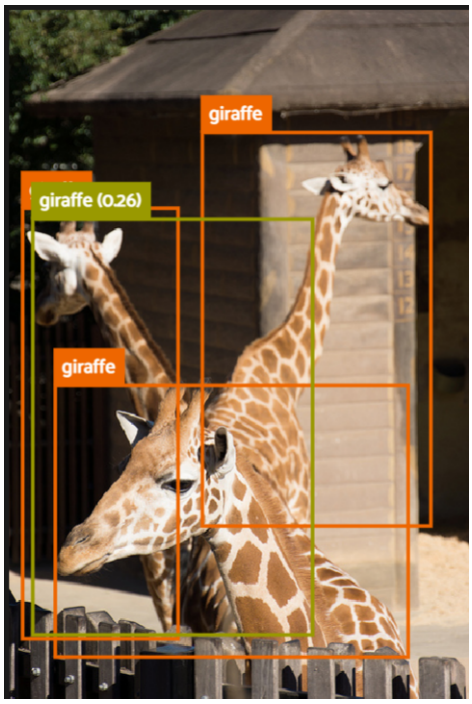
- The model is better at detecting animals/people than detecting objects, maybe because animals/people have different and unique shapes and sizes, while objects can easily be mixed up.

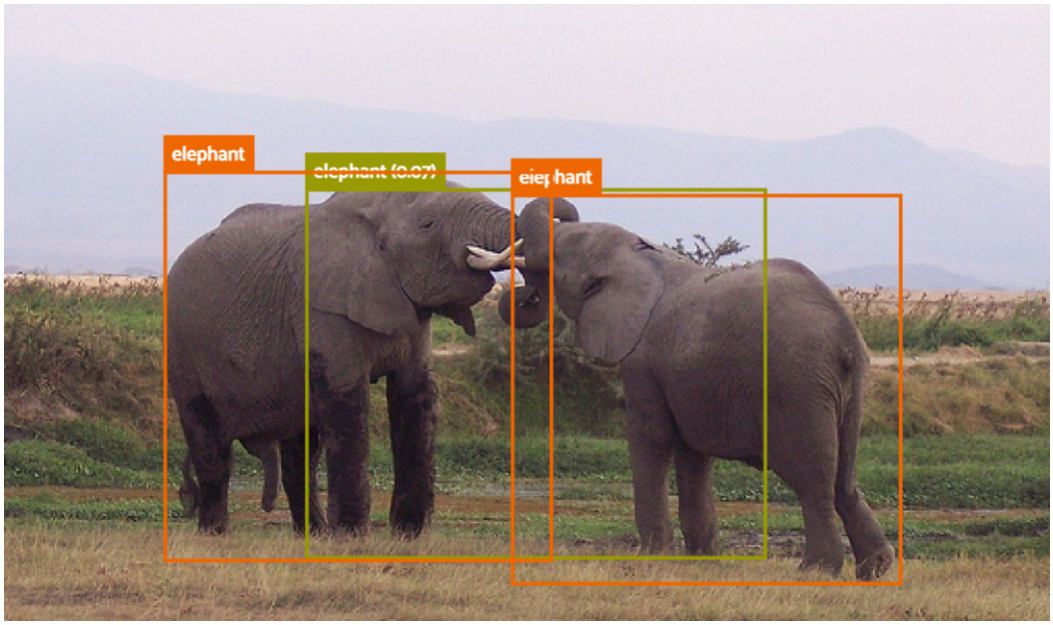
Failure cases:




person
sports ball (0.07)
person
sports ball
tenni racket
tennis racket (0.16)


person (0.07)
cat
cat (0.09)
tv (0.12)
bed (0.19)
ouch (0.09)
bed


giraffe
giraffe (0.26)
giraffe


elephant

Comments:

- All these results have confidence of 0.3 or less.

- From the results, we can notice that when an object has a part of it hidden and appears again, it's detected as 2 different parts.

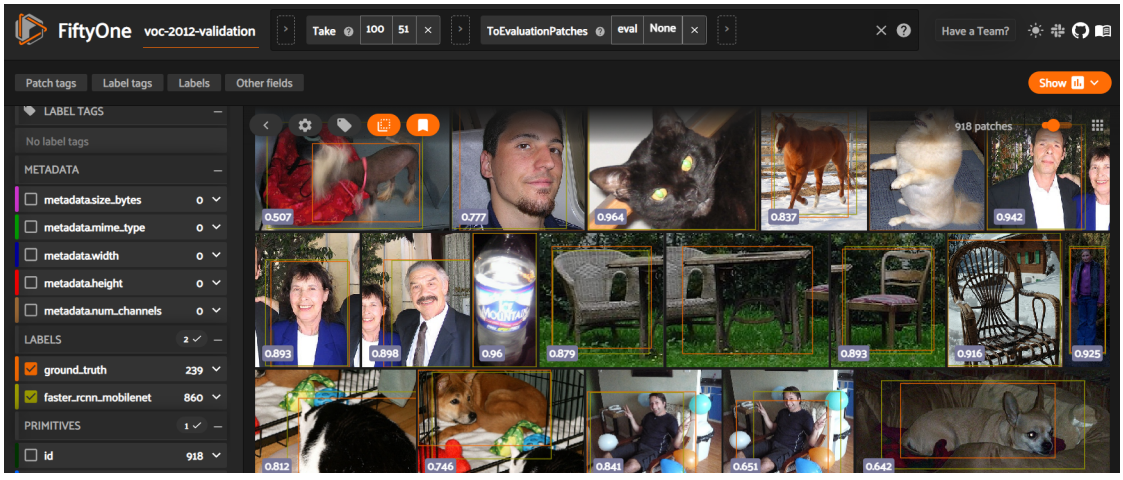
- The model has mistaken the head to be a sports ball as they both have the same shape (round), however they both have different sizes so maybe it's not good with differentiating sizes.

- The model has mistaken the laptop for TV, which means, as we mentioned earlier, that it finds it hard to detect different objects and work on shapes more than sizes or maybe it just doesn't have a "laptop" class so it detects it as the nearest object to it.

- The model seems to fail at detecting edges, where things start and begin, because as we can see in the elephants image, it detected an elephant by having a bounding box in the middle of the two elephants and same with the giraffes.

4.2.2 VOC 2012 Dataset

Here, we'll evaluate the results obtained when testing the model on VOC 2012 Dataset, validation split.
IoU:



Mean Average Precision (mAP):

mAP of this model on VOC 2012

```
[ ]  print(results_voc2012.mAP())

     0.3161495509323367
```

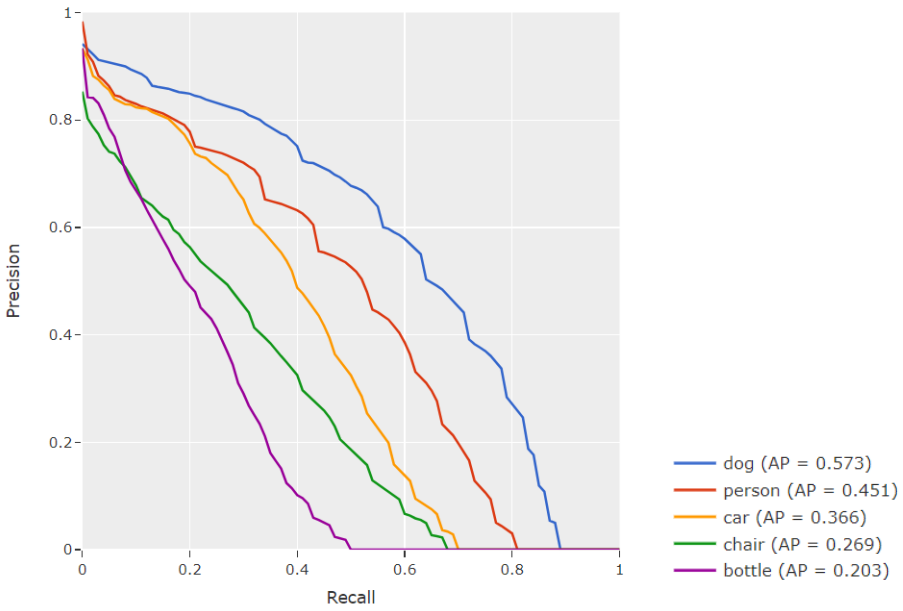
True positives, false positives and false negatives count:

```
View stages:
    1. Take(size=100, seed=51)
    2. ToEvaluationPatches(eval_key='eval', config=None)
{'tp': 181, 'fn': 58, 'fp': 679}
```

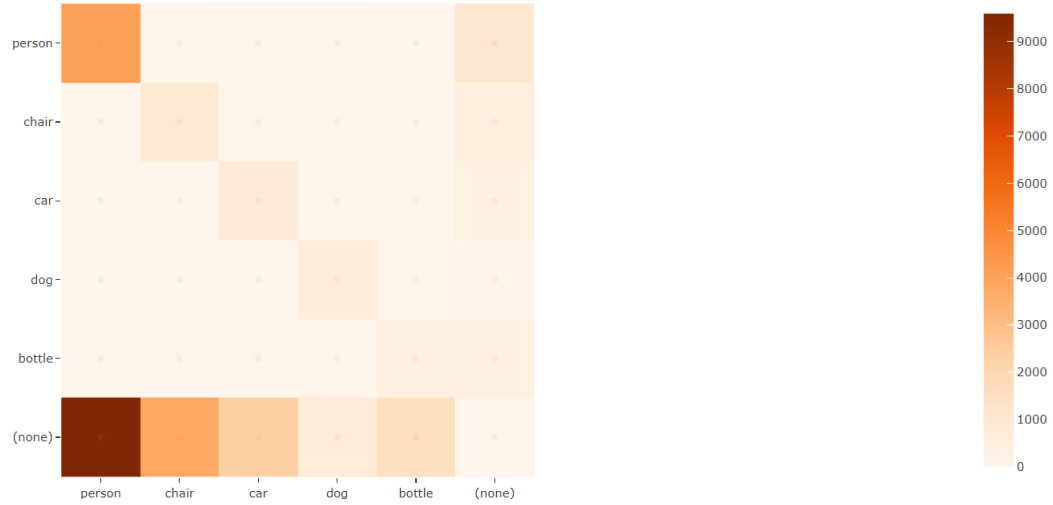
Top 5 most common classes classification report:

	precision	recall	f1-score	support
person	0.30	0.80	0.44	5110
chair	0.21	0.67	0.32	1449
car	0.26	0.70	0.37	1173
dog	0.47	0.88	0.62	773
bottle	0.18	0.49	0.27	733
micro avg	0.28	0.75	0.40	9238
macro avg	0.28	0.71	0.40	9238
weighted avg	0.28	0.75	0.41	9238

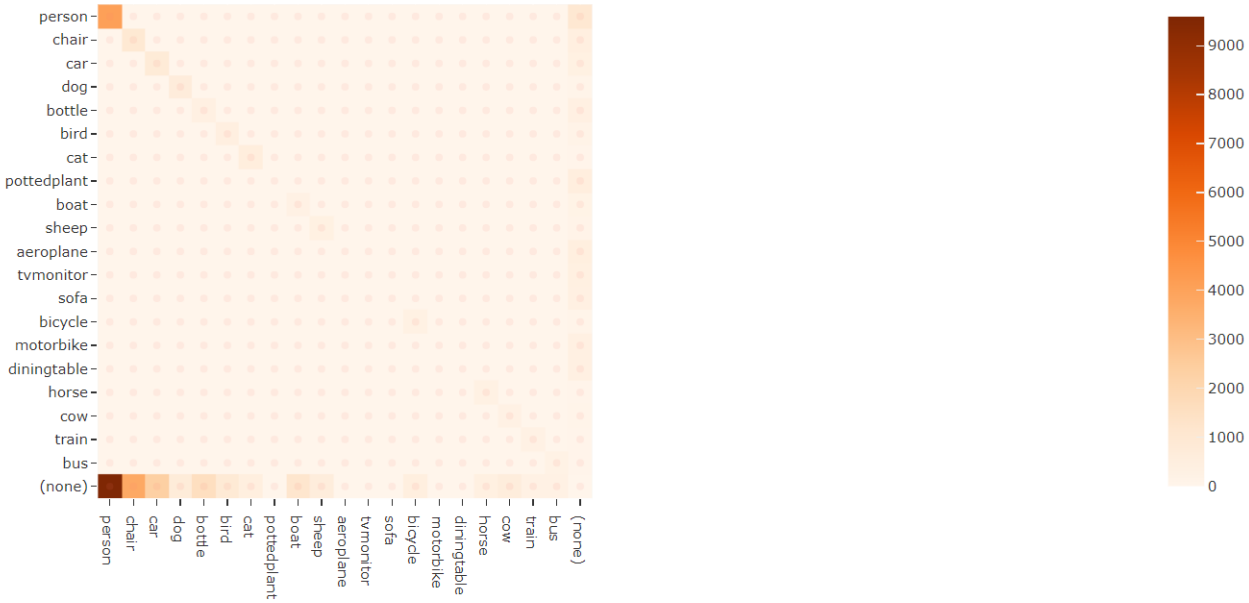
Precision vs. Recall for top 5 classes:



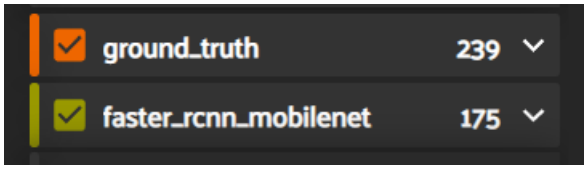
Confusion matrix for top 5 classes:



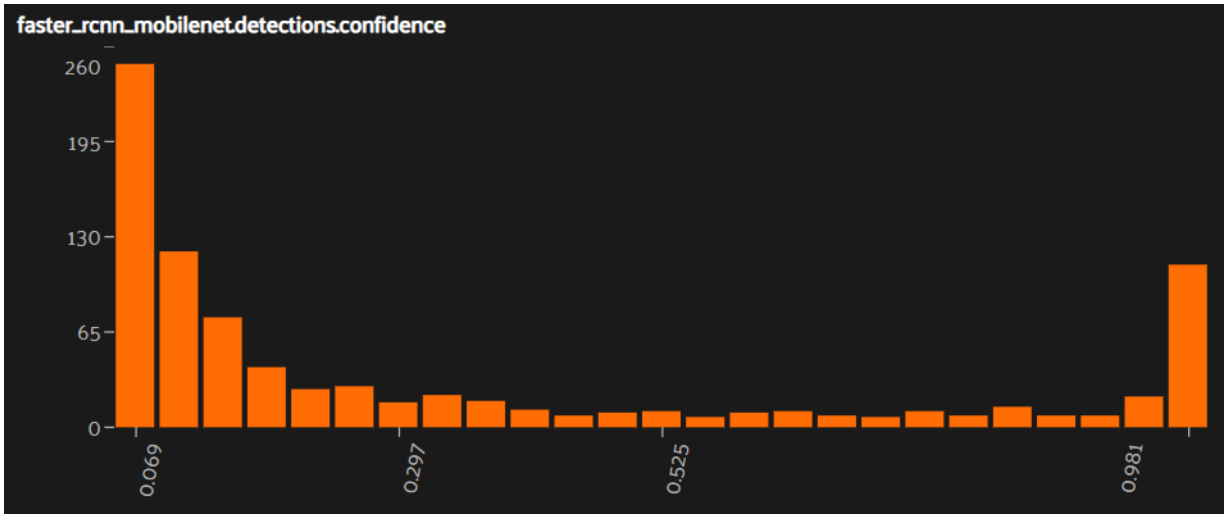
Confusion matrix for all 20 classes:



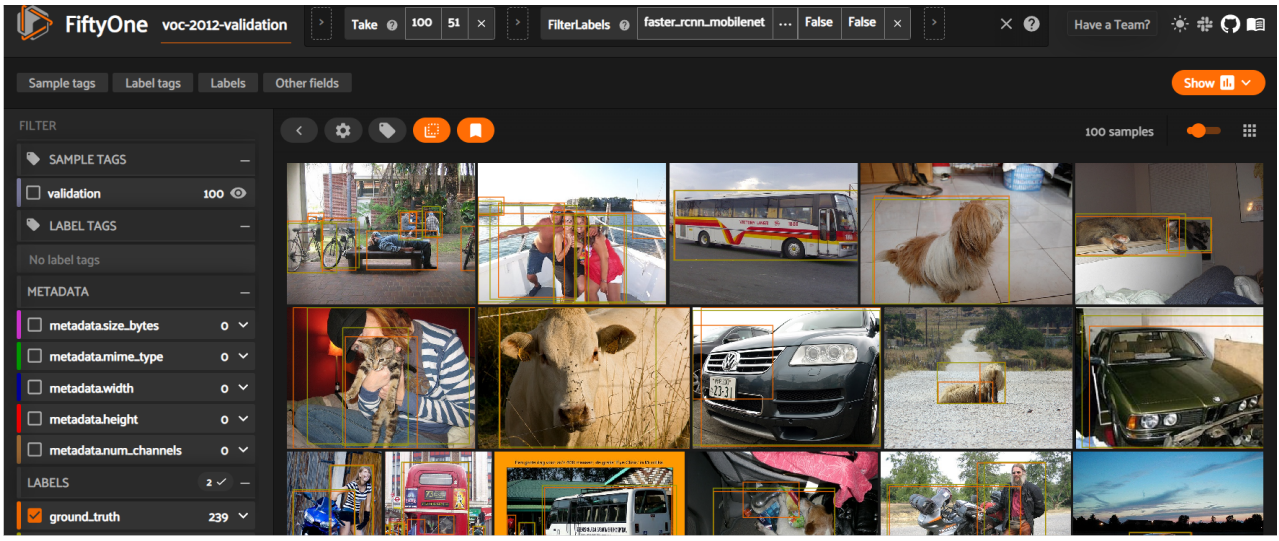
Number of actual objects in 100 samples vs number of objects detected by model:

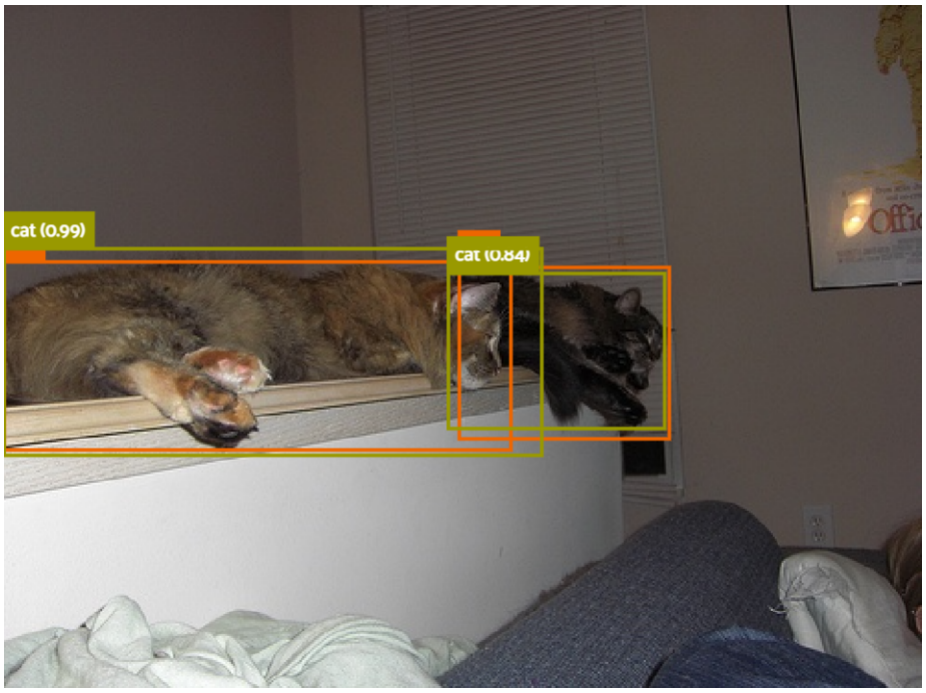


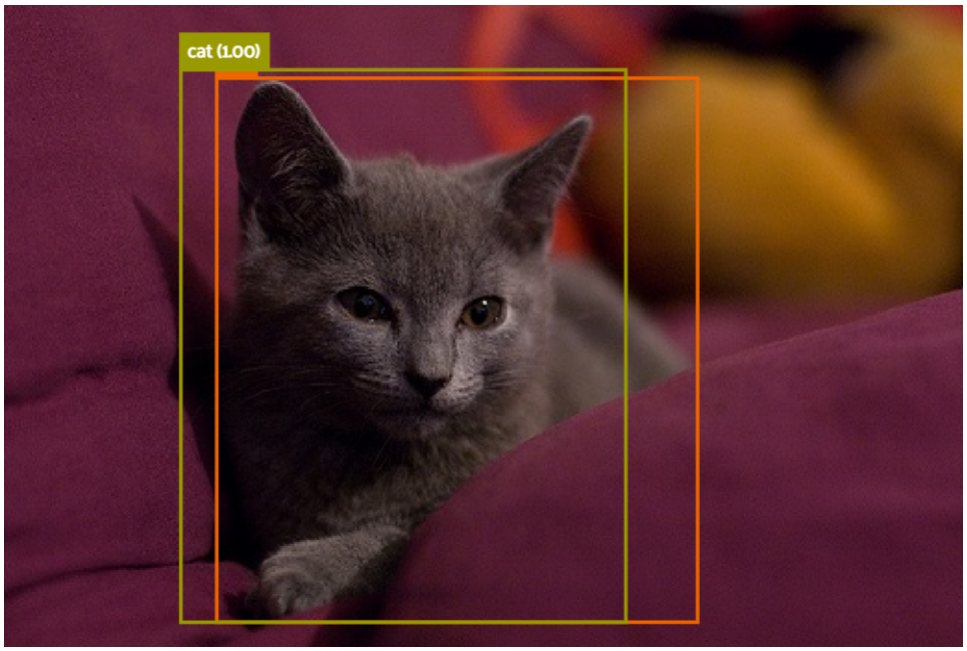
Confidence level:

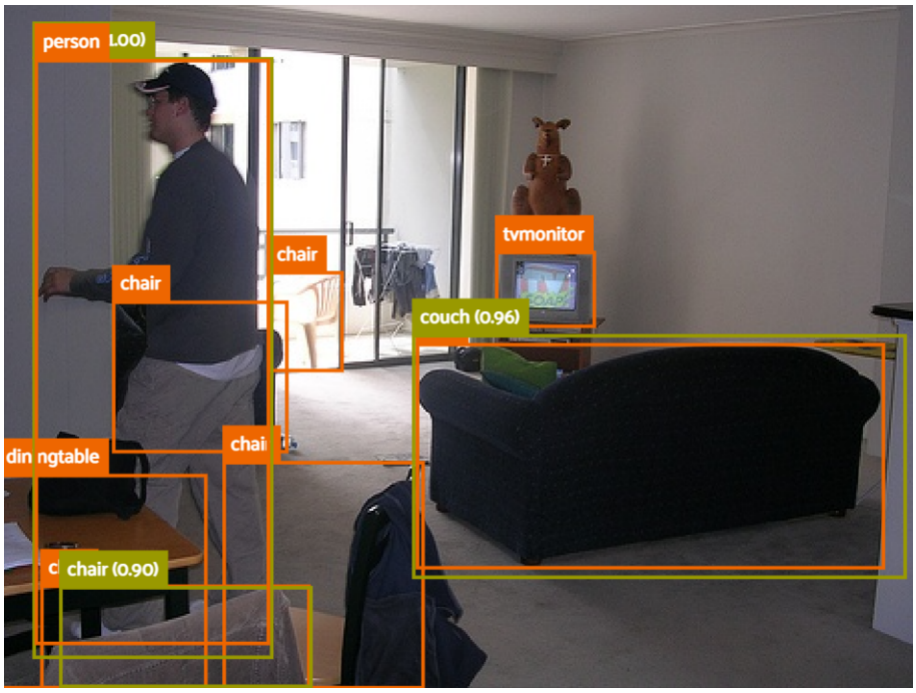


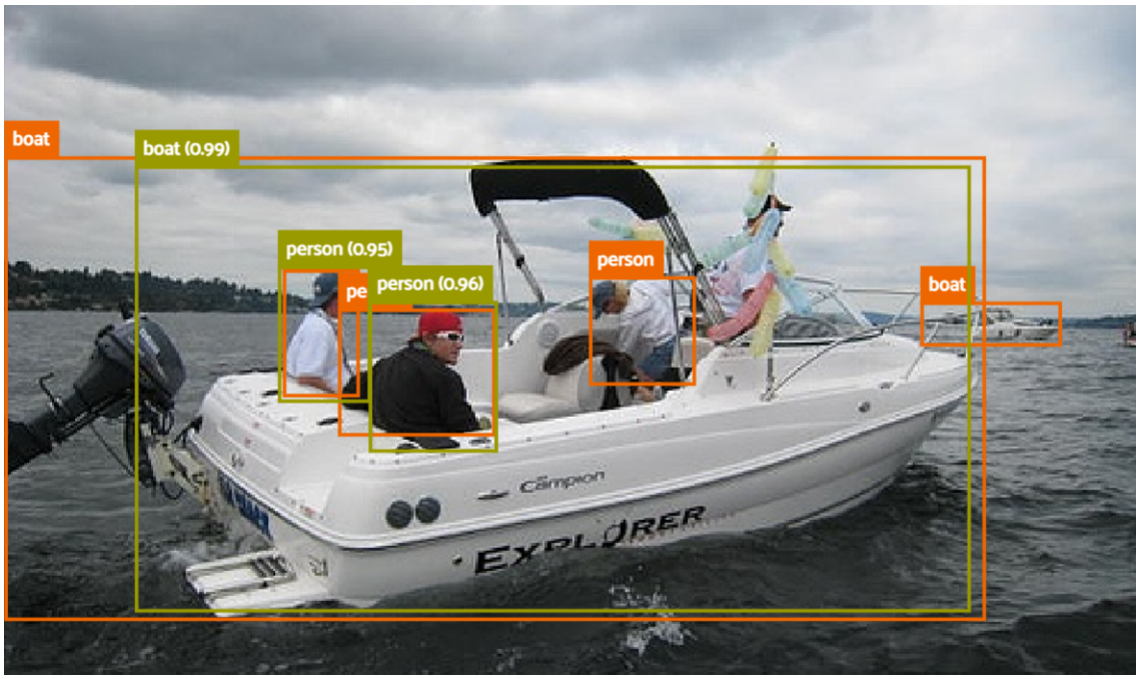
Success cases:




cat (0.99)
cat (0.84)


cat (1.00)


person
chair
chair
tvmonitor
couch (0.96)
chai
ngtable
chair (0.90)


boat
boat (0.99)
person (0.95)
person (0.96)
person
boat
EXPLORER

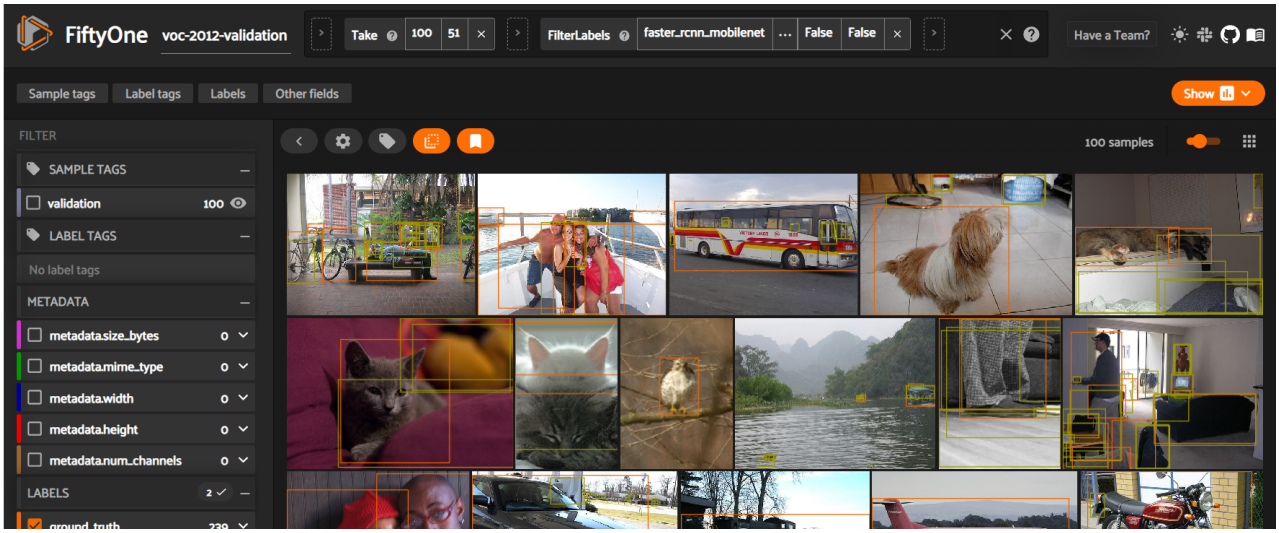
Comments:

- All these results have confidence of 0.75 or more.

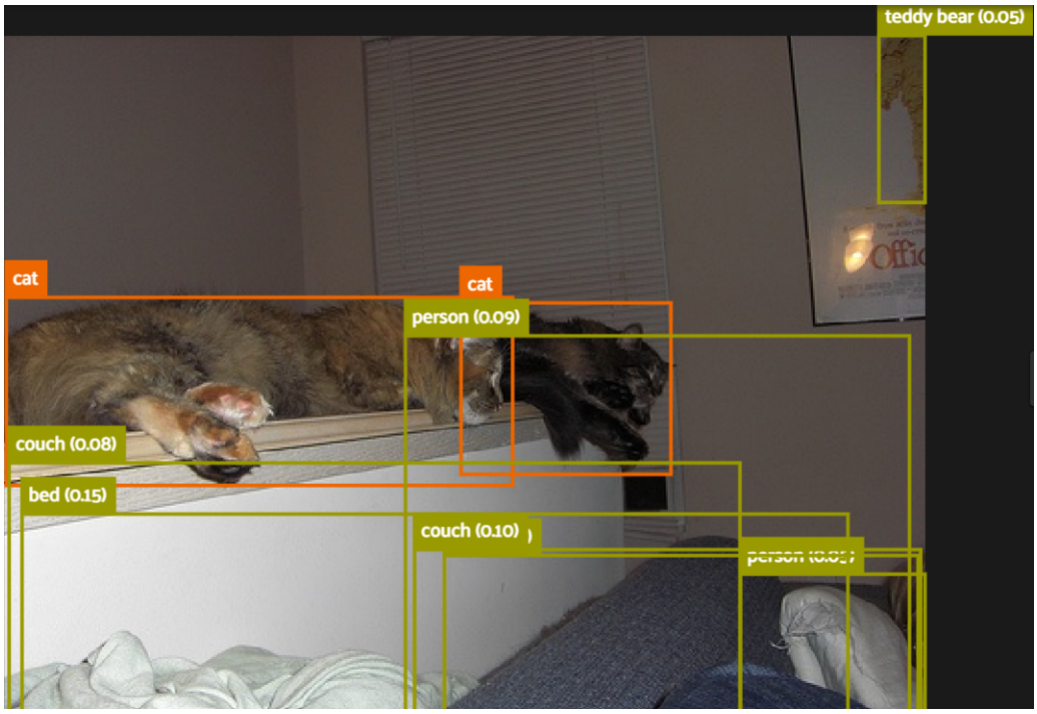
- This model is relatively good at detecting objects/animals/people even when parts of them are hidden on this dataset, maybe because it has fewer number of classes so variety of objects is limited.

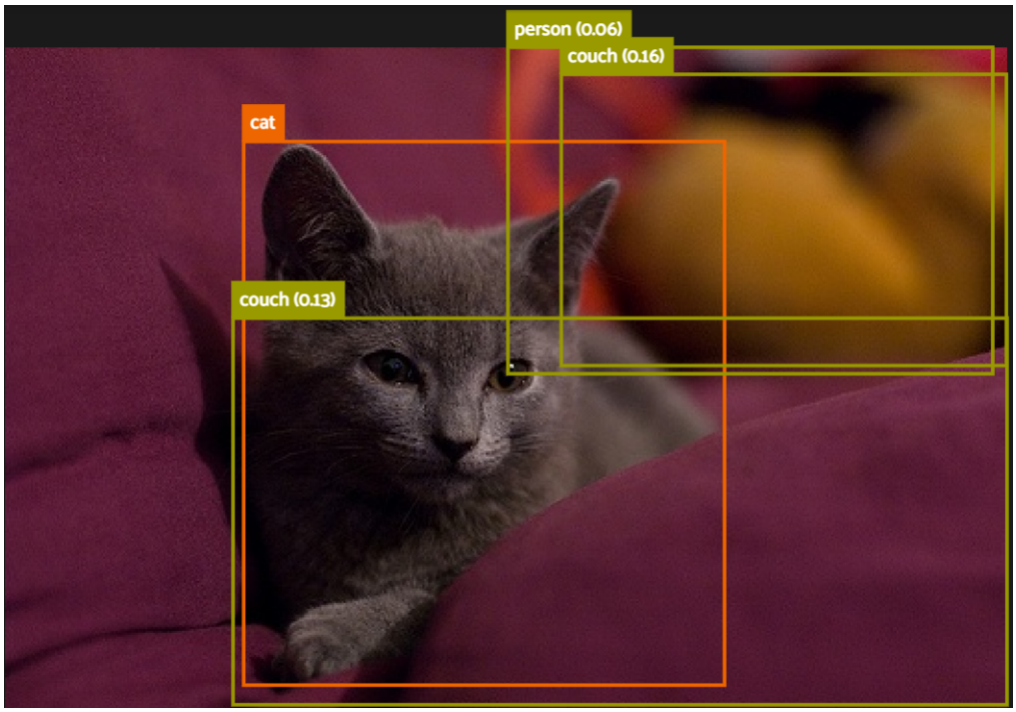
- It can detect things that are at a far distant correctly in this dataset.

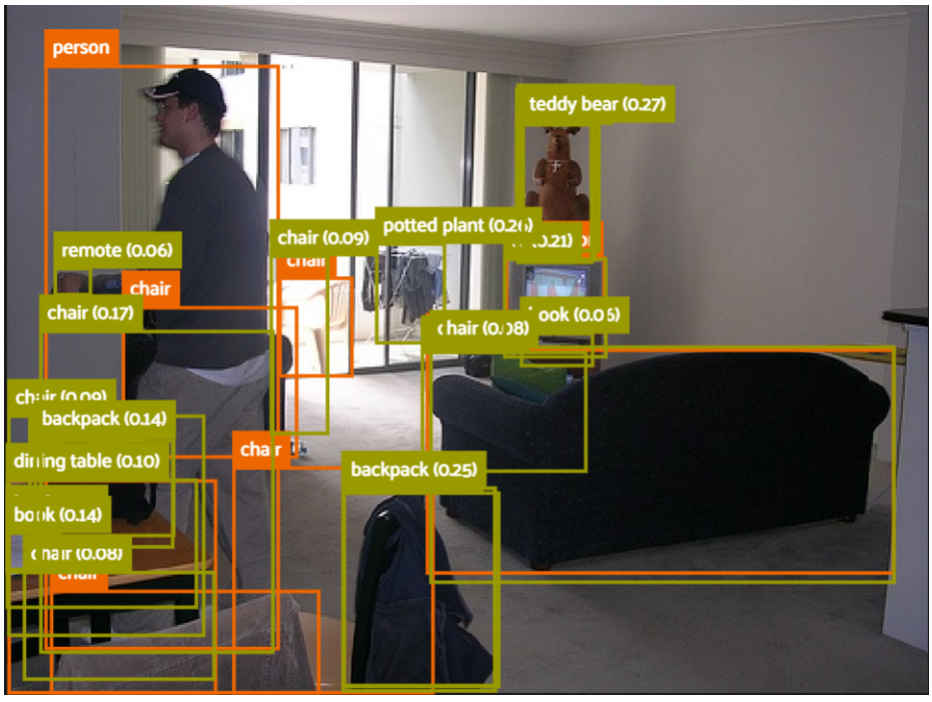
Failure cases:

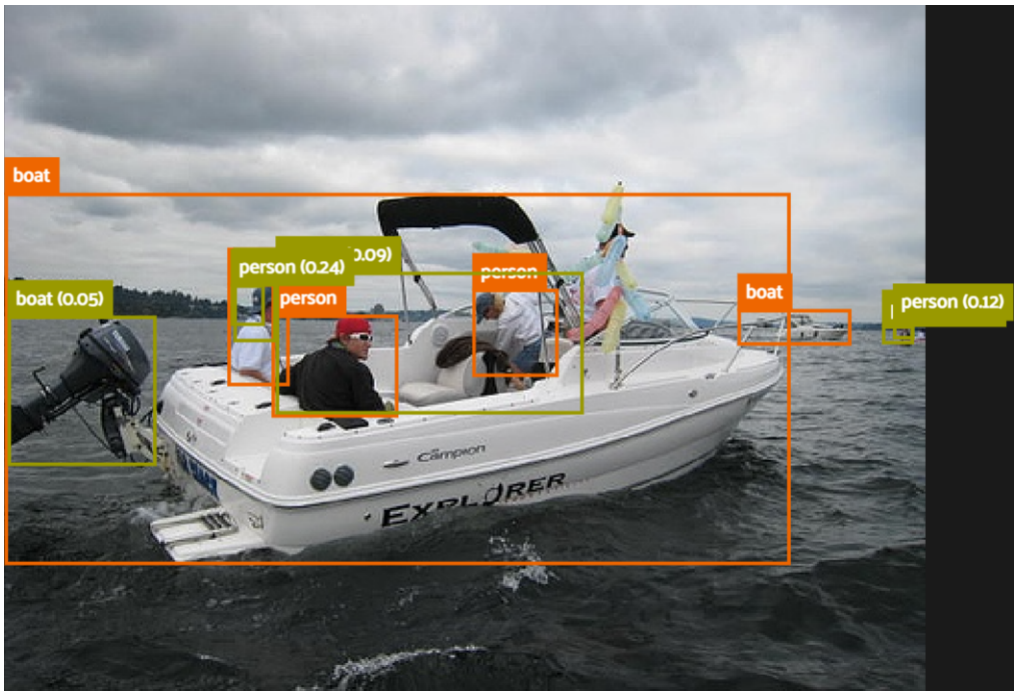



bus
person (0.11)
person (0.08)
faster_rcnn_mobilenet
person
label
0.068
confidence
fp
eval
eval_id
63a58daa1f385b975c6883c8
id
VICTORY LINER
1888


teddy bear (0.05)
cat
cat
person (0.09)
couch (0.08)
bed (0.15)
couch (0.10)


person (0.06)
couch (0.16)
cat
couch (0.13)


person
teddy bear (0.27)
chair (0.09)
potted plant (0.20)
remote (0.06)
chair
chair (0.17)
backpack (0.14)
backpack (0.25)



Comments:

- All these results have confidence of 0.3 or less.

- In this dataset, it has mistaken parts of objects as people.

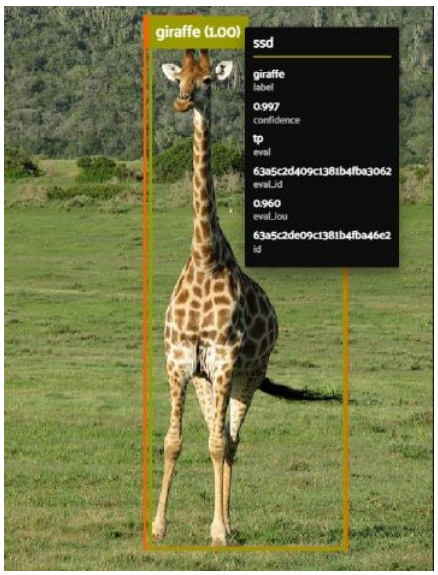
- As this dataset has fewer number of classes than COCO, the model here detects unnecessary objects in the image that are mostly wrong.

- Here, the model has mistaken parts attached to the object as the object itself, probably because the limited number of classes again.

4.3 Model (3)

4.3.1 COCO Dataset 2017

Here, we'll evaluate the results obtained when testing the model on COCO Dataset 2017, validation split.
IoU:



Mean Average Precision (mAP):

▾ mAP of this model

```
[ ]  print(results.mAP())

     0.21451017794591154
```

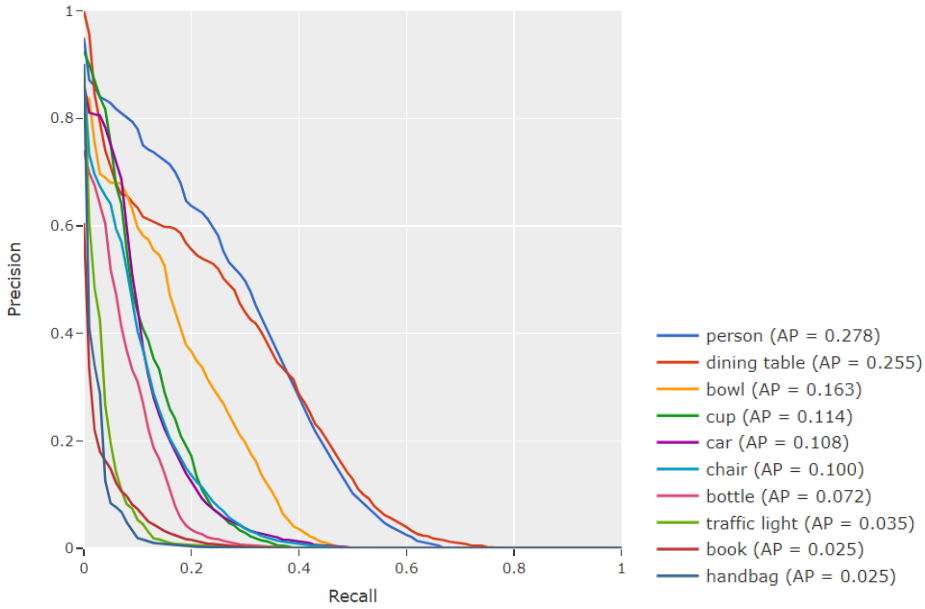
True positives, false positives and false negatives count:

```
View stages:
    1. Take(size=100, seed=51)
    2. ToEvaluationPatches(eval_key='eval', config=None)
{'tp': 468, 'fn': 295, 'fp': 28858}
```

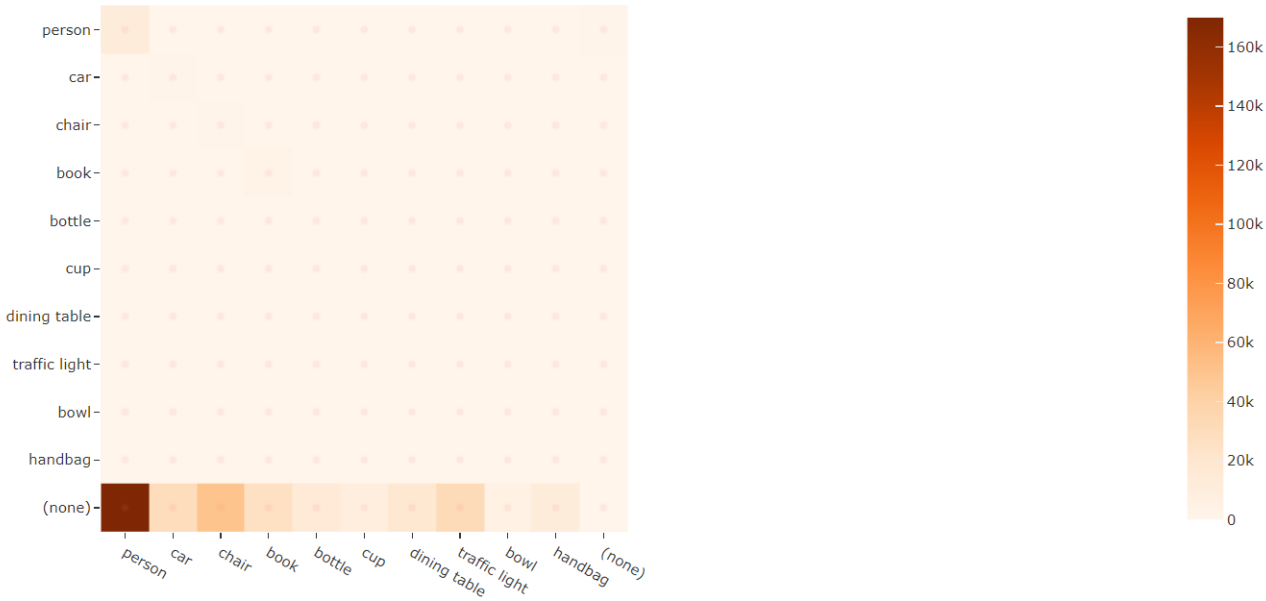
Top 10 most common classes classification report:

	precision	recall	f1-score	support
person	0.07	0.88	0.12	13902
car	0.03	0.63	0.06	1438
chair	0.02	0.67	0.03	1214
book	0.07	0.86	0.14	2513
bottle	0.02	0.50	0.04	650
cup	0.03	0.50	0.06	593
dining table	0.01	0.77	0.03	329
traffic light	0.01	0.48	0.01	429
bowl	0.02	0.47	0.05	312
handbag	0.01	0.29	0.01	255
micro avg	0.04	0.80	0.08	21635
macro avg	0.03	0.61	0.05	21635
weighted avg	0.06	0.80	0.11	21635

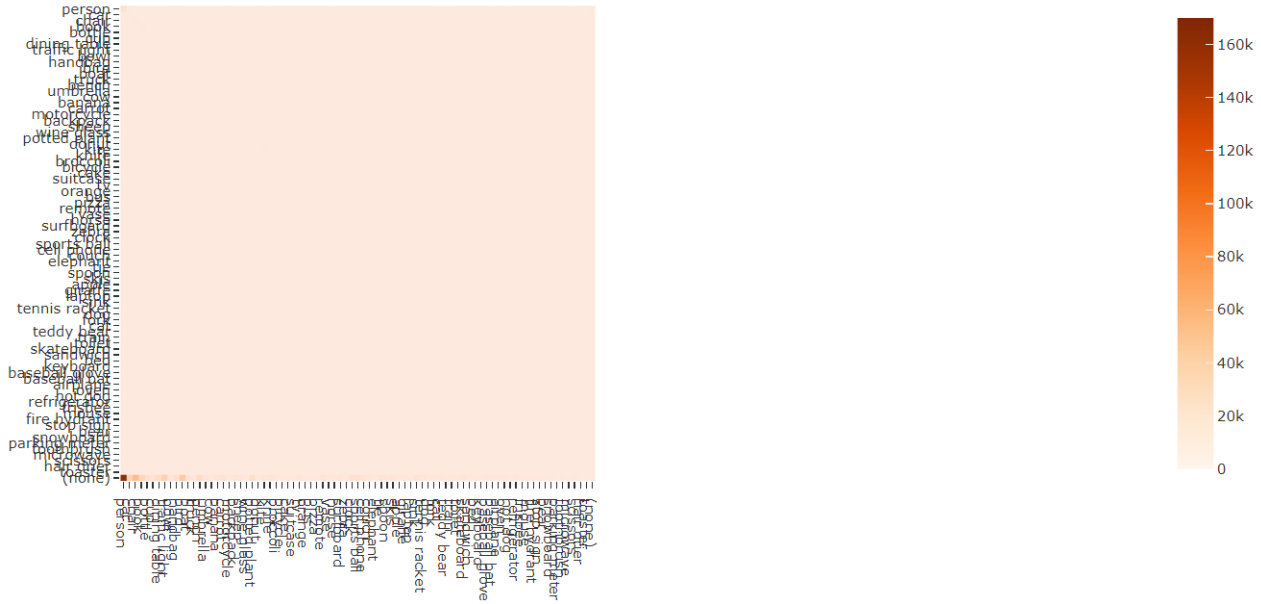
Precision vs. Recall for top 10 classes:



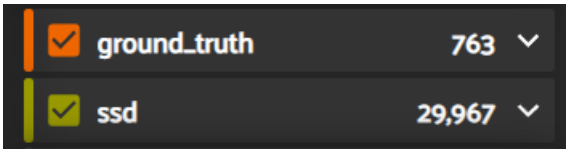
Confusion matrix for top 10 classes:



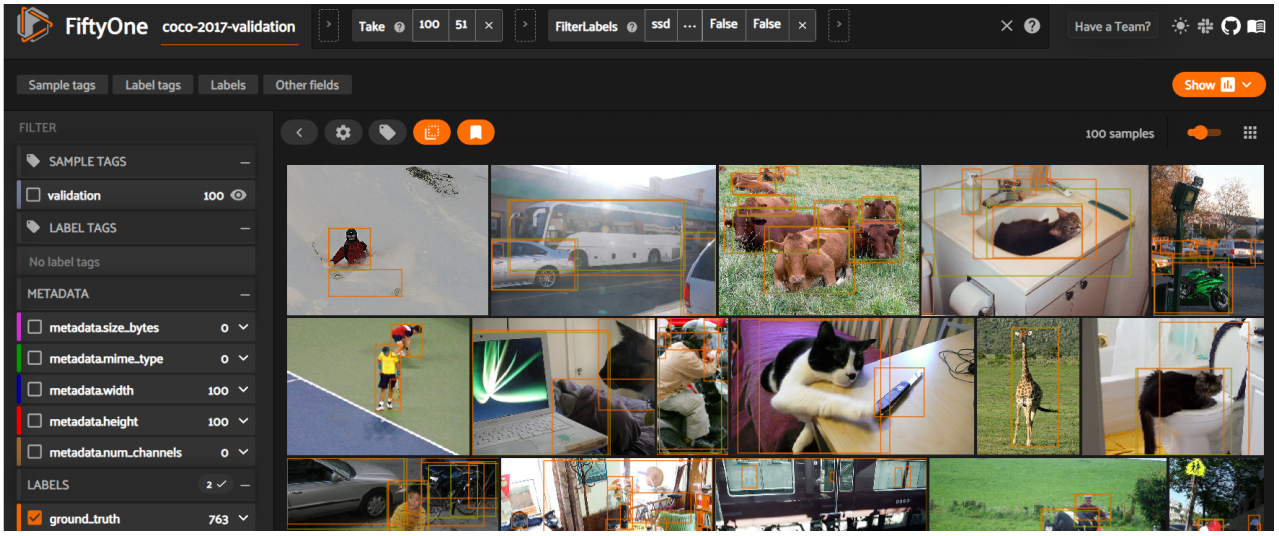
Confusion matrix for all 80 classes:

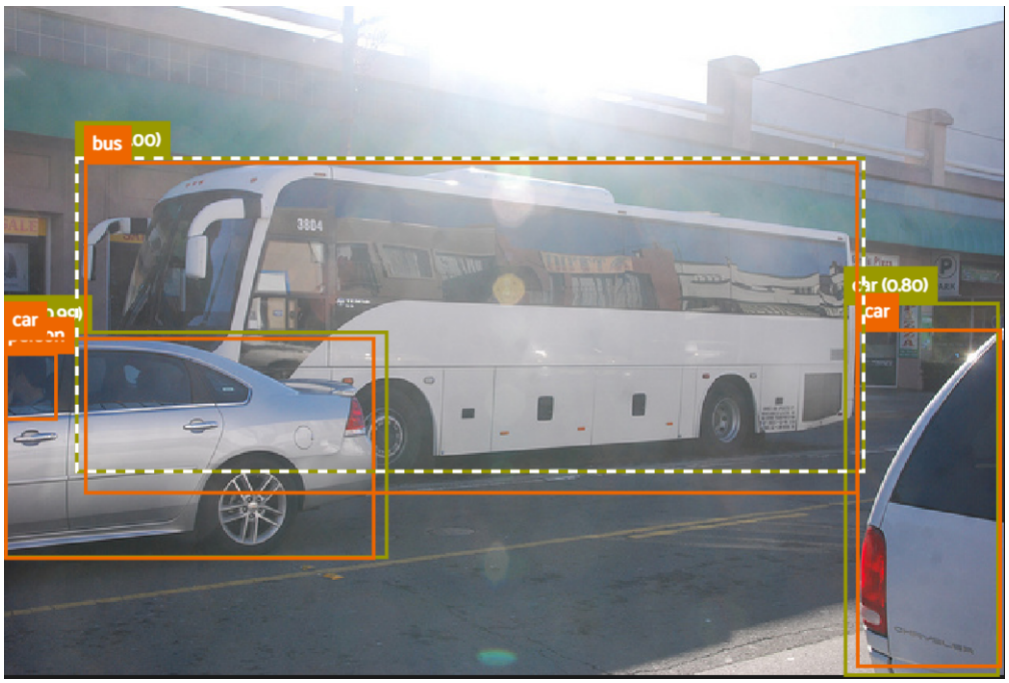


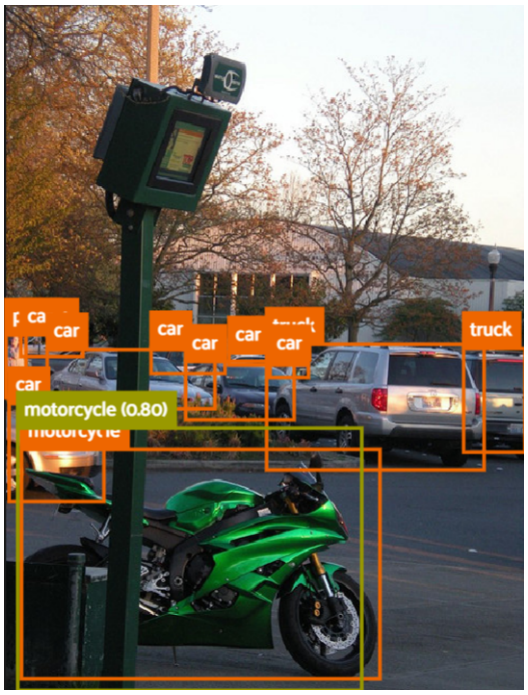
Number of actual objects in 100 samples vs number of objects detected by model:

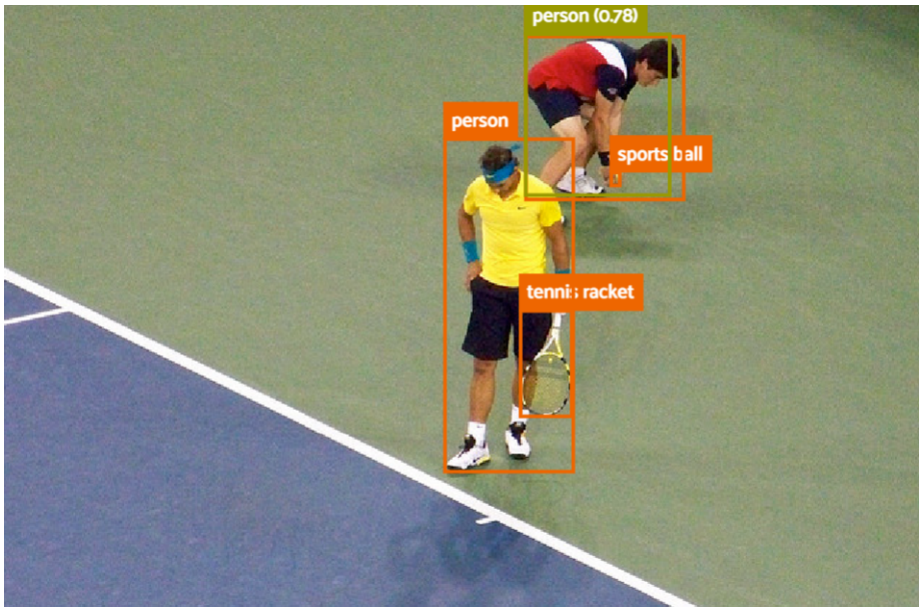


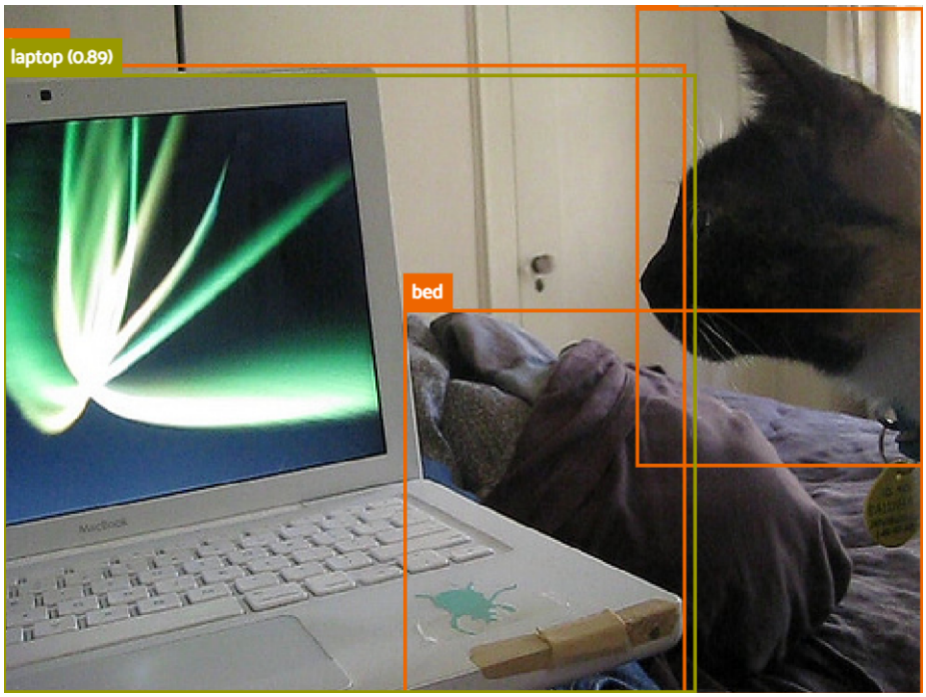
Success cases:

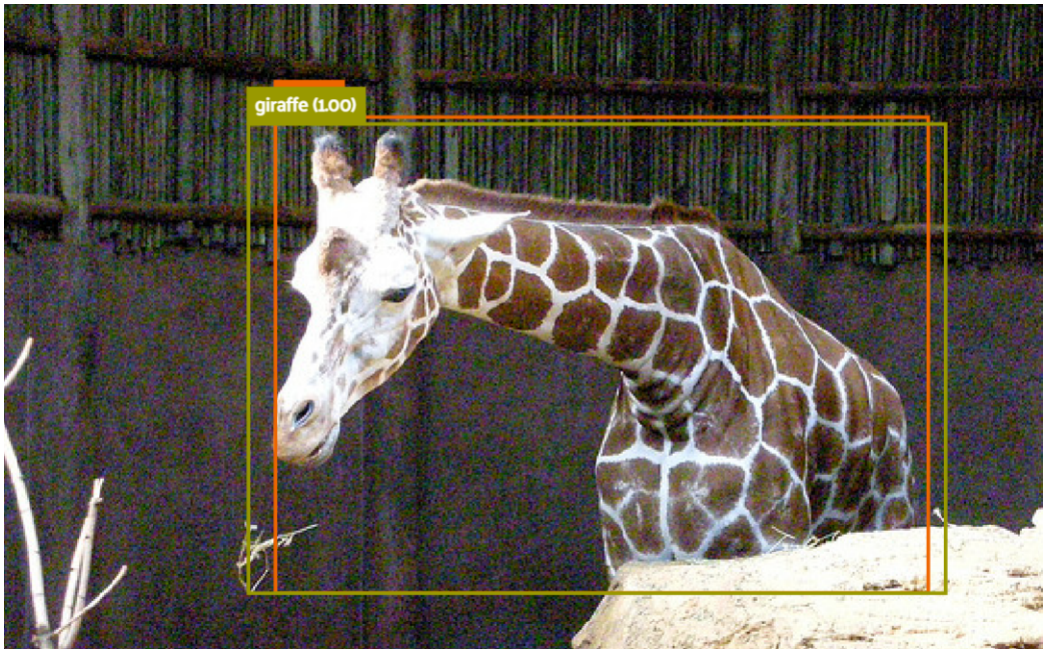





car
car
car
car
car
truck
motorcycle (0.80)


person (0.78)
person
sports ball
racket


laptop (0.89)
bed

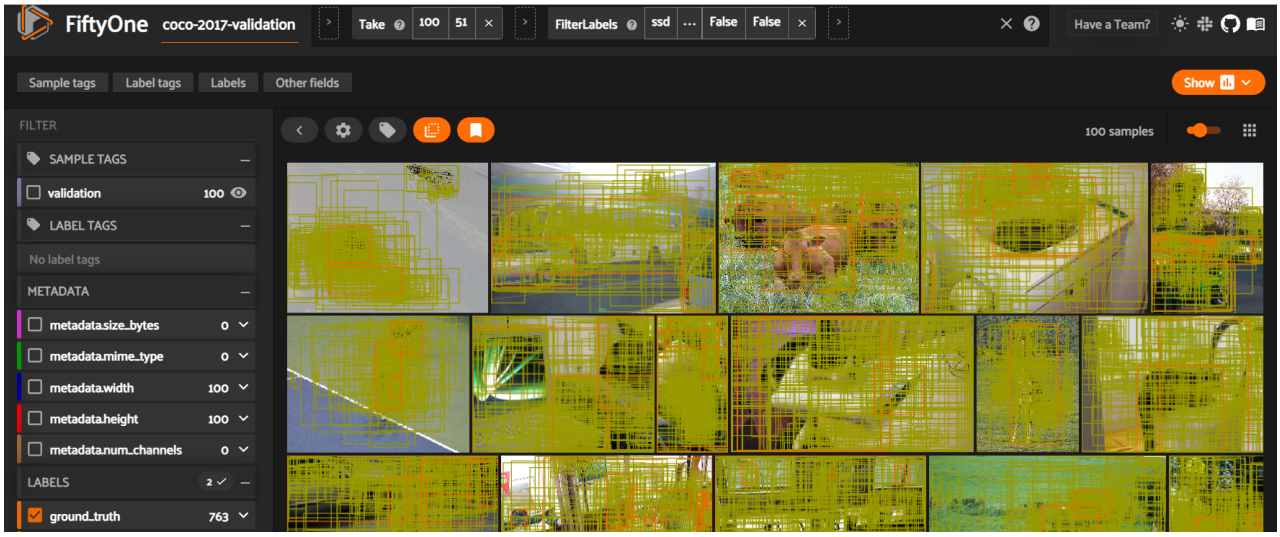

giraffe (1.00)

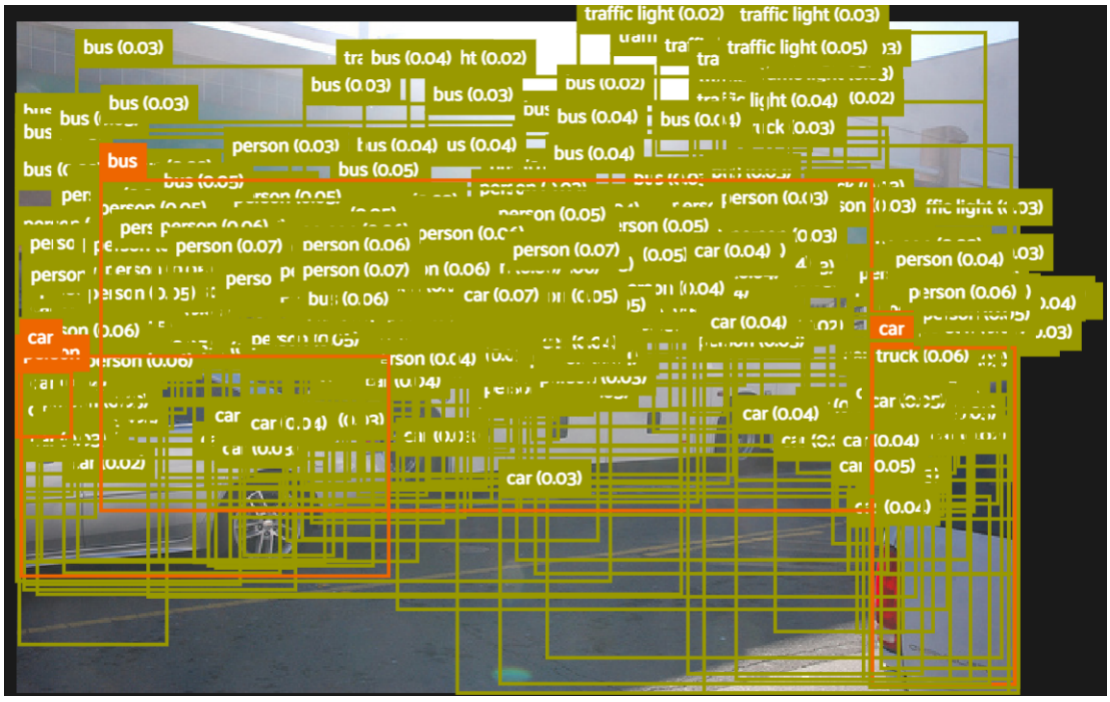
Comments:

- All these results have confidence of 0.75 or more.

- This model detects objects well but nit all of them. Many objects in the images pass by it, and the ones detected relatively don't have very high confidence.

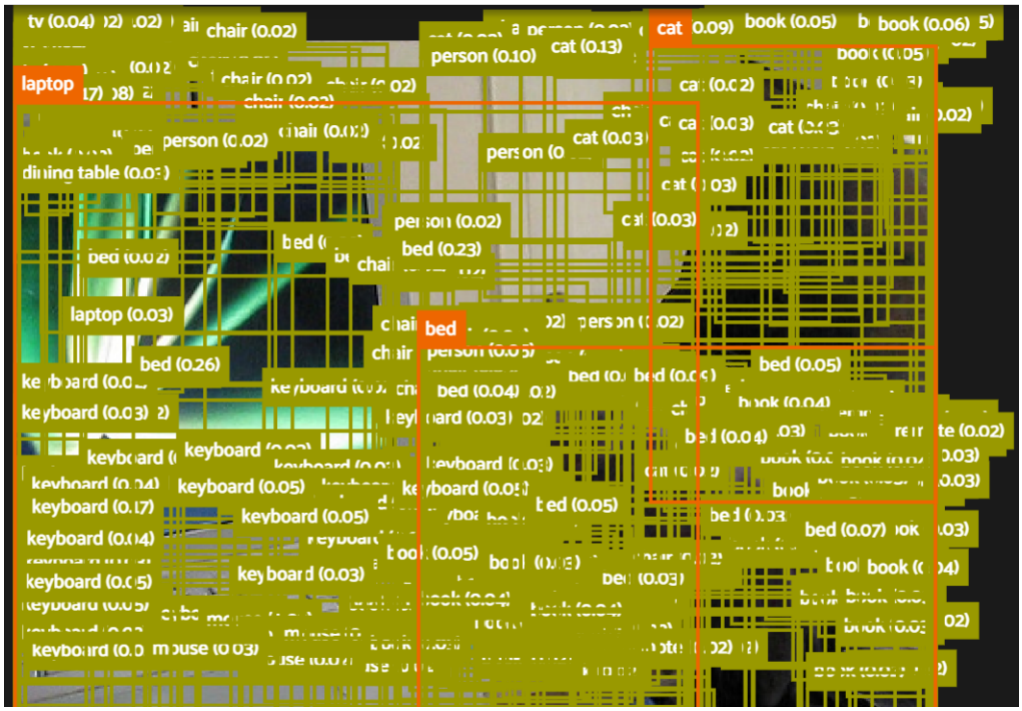
Failure cases:




traffic light (0.02)
traffic light (0.03)
bus (0.03)
bus (0.04)
traffic light (0.05)
bus (0.03)
bus (0.03)
bus (0.02)
bus (0.03)
traffic light (0.04)
(0.02)
bus (0.04)
bus (0.04)
person (0.03)
bus (0.04)
bus (0.04)
bus
bus (0.05)
person (0.03)
person (0.05)
person (0.05)
person (0.07)
person (0.06)
person (0.06)
person (0.07)
car (0.04)
person (0.04)
person (0.07)
car (0.07)
person (0.06)
car (0.04)
car
truck (0.06)
car (0.04)
car (0.04)
car (0.05)
car (0.03)


traffic light (0.05)
traffic light (0.06)
traffic light (0.14)
traffic light (0.07)
traffic light (0.05)
traffic light (0.07)
truck (0.08)
traffic light (0.08)
traffic light (0.07)
car (0.20)
car
car
car
truck
car (0.07)
car (0.11)
motorcycle
motorcycle (0.13)
motorcycle (0.13)


person
sports ball
tennis racket


laptop
cat
bed
dining table

Comments:

- All these results have confidence of 0.3 or less.

- From the results, we can notice that this model generates way too many bounding boxes and most of them are wrong.

- The output images can hardly be seen from the amount of bounding boxes predicted and after filtering them, very few actually detects the objects correctly.

- This huge amount of bounding boxes takes much more time to compute and they don't even yield good result in comparison to other models.

4.3.2 VOC 2012 Dataset

Here, we'll evaluate the results obtained when testing the model on VOC 2012 Dataset, validation split.
IoU:

Mean Average Precision (mAP):

mAP of this model on VOC 2012

```
[ ]  print(results_voc2012.mAP())
```

```
0.3177410338420653
```

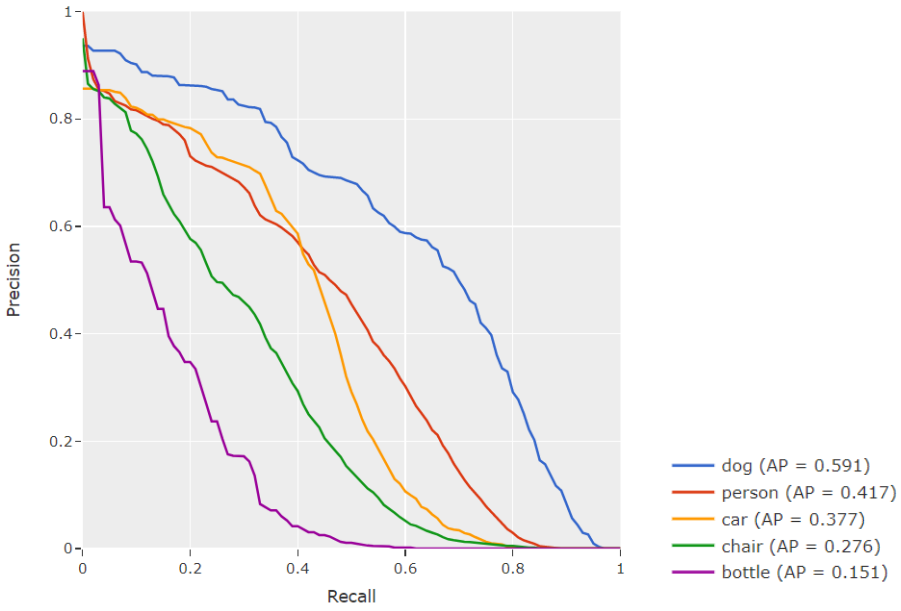
True positives, false positives and false negatives count:

```
 View stages:
    1. Take(size=100, seed=51)
    2. ToEvaluationPatches(eval_key='eval', config=None)
{'fp': 29802, 'tp': 198, 'fn': 41}
```

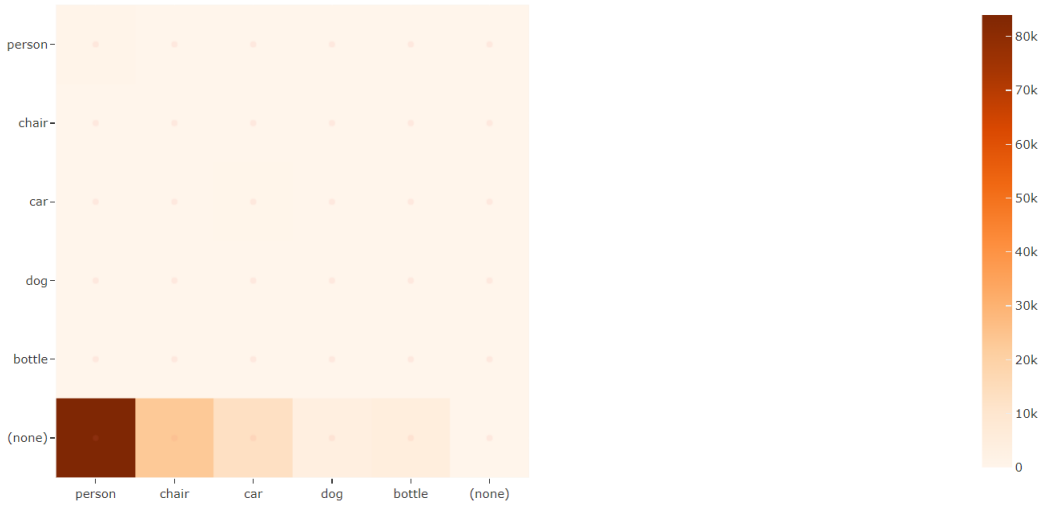
Top 5 most common classes classification report:

	precision	recall	f1-score	support
person	0.01	0.89	0.02	921
chair	0.01	0.85	0.02	209
car	0.01	0.81	0.03	237
dog	0.03	0.96	0.06	139
bottle	0.01	0.61	0.02	80
micro avg	0.01	0.86	0.02	1586
macro avg	0.01	0.82	0.03	1586
weighted avg	0.01	0.86	0.02	1586

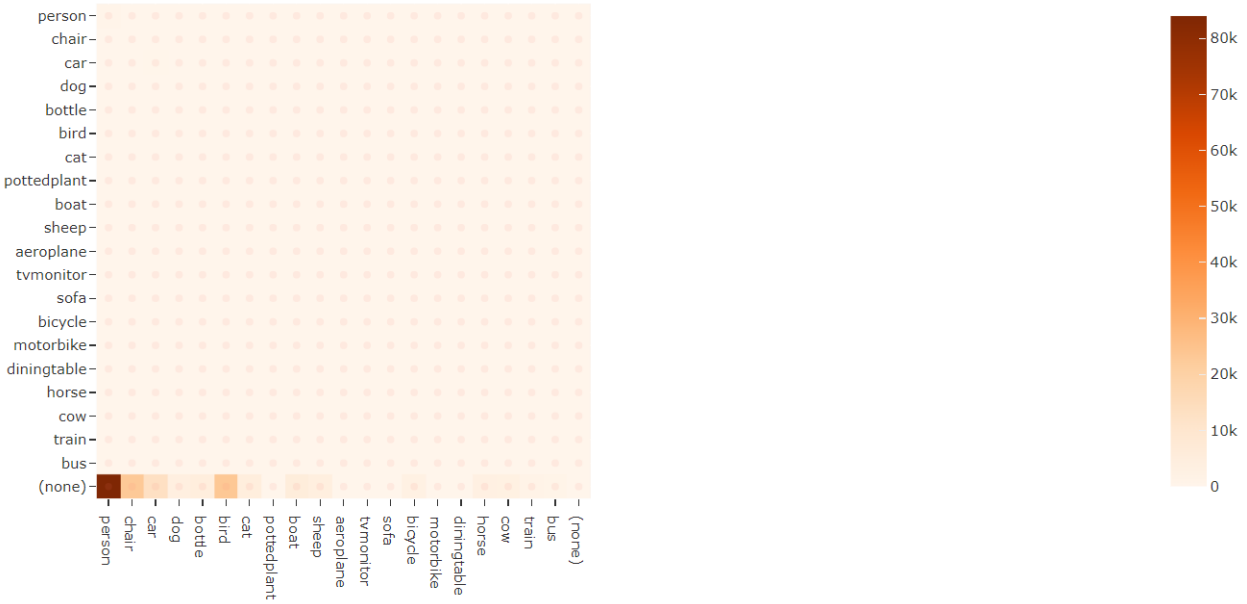
Precision vs. Recall for top 5 classes:



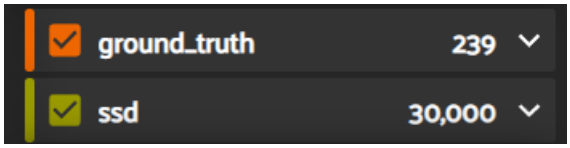
Confusion matrix for top 5 classes:



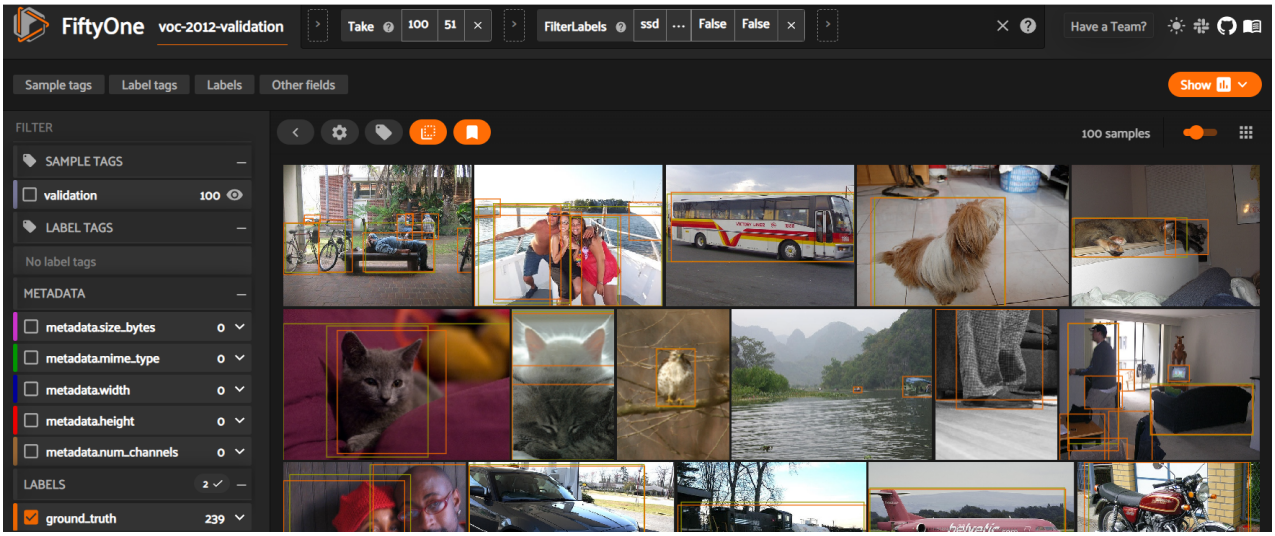
Confusion matrix for all 20 classes:

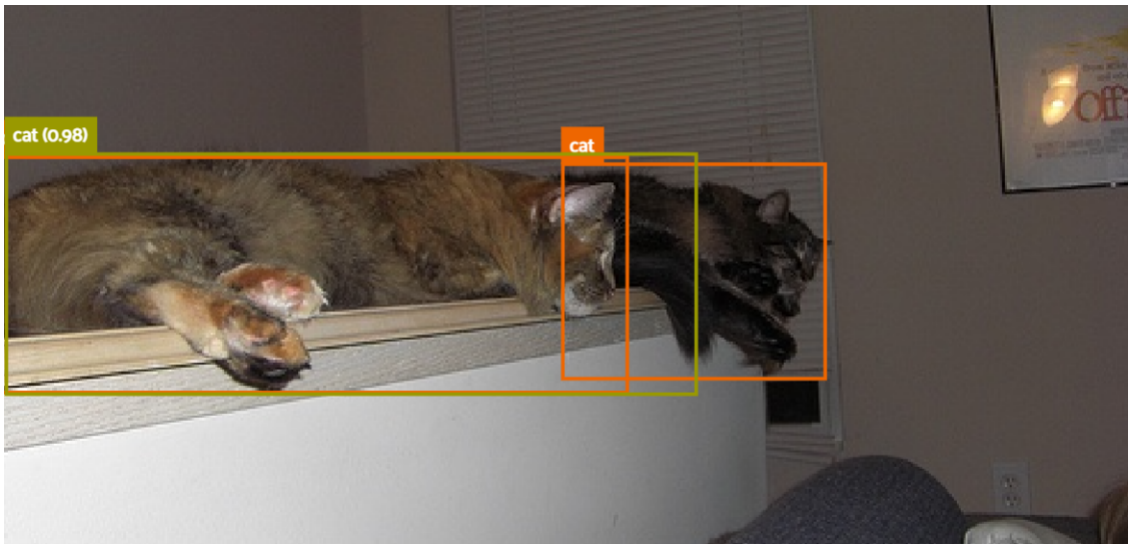


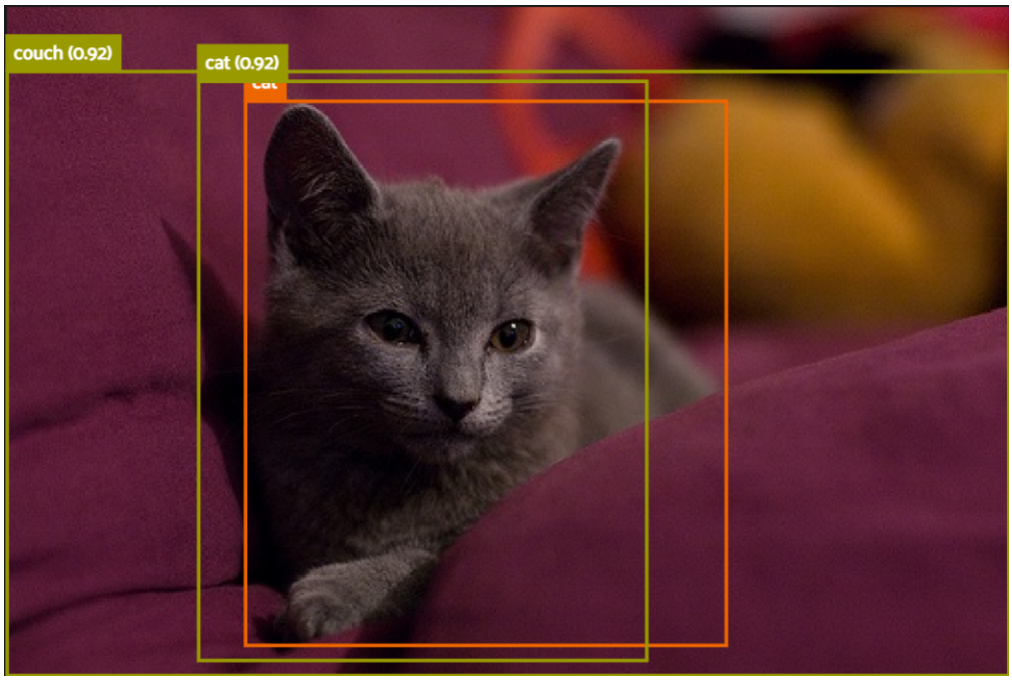
Number of actual objects in 100 samples vs number of objects detected by model:

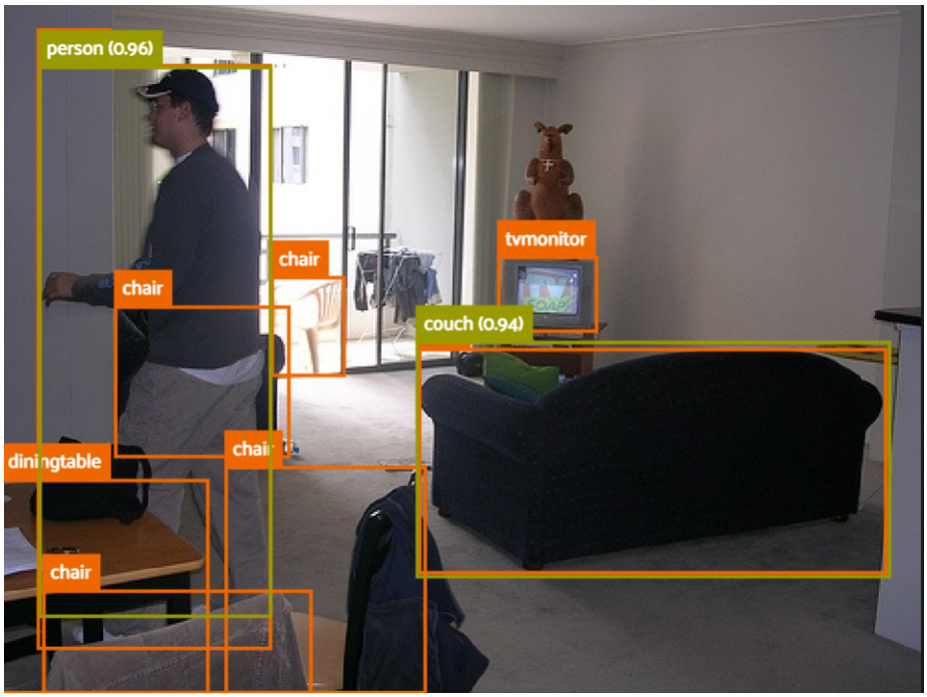


Success cases:




cat (0.98)
cat


couch (0.92)
cat (0.92)


person (0.96)
chair
tvmonitor
chair
couch (0.94)
chai
chair

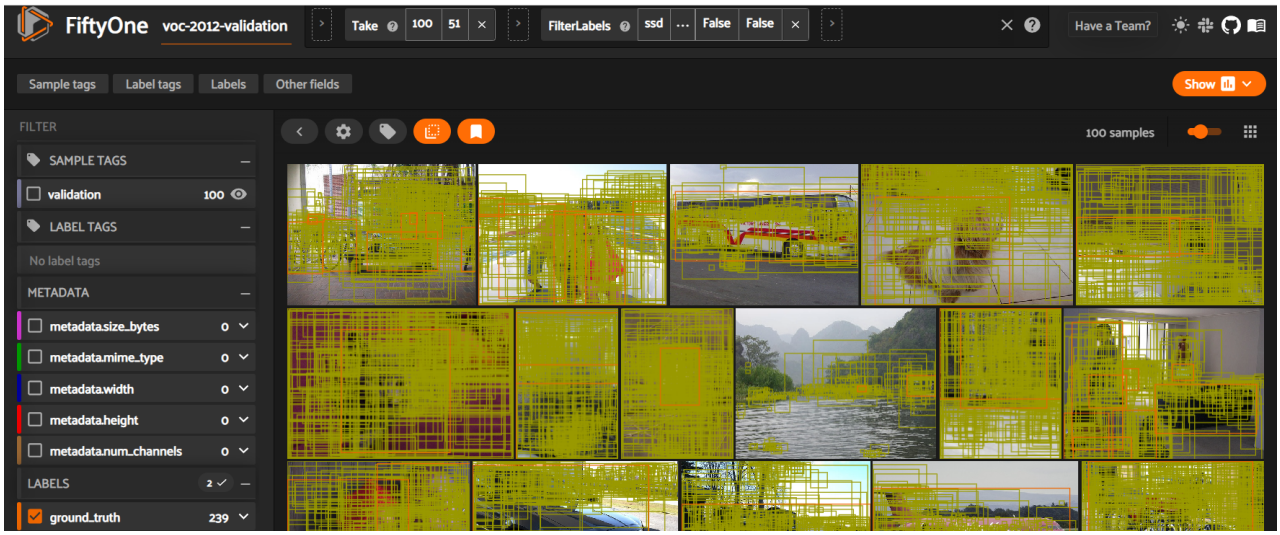

airplane (1.00)
Lufthansa

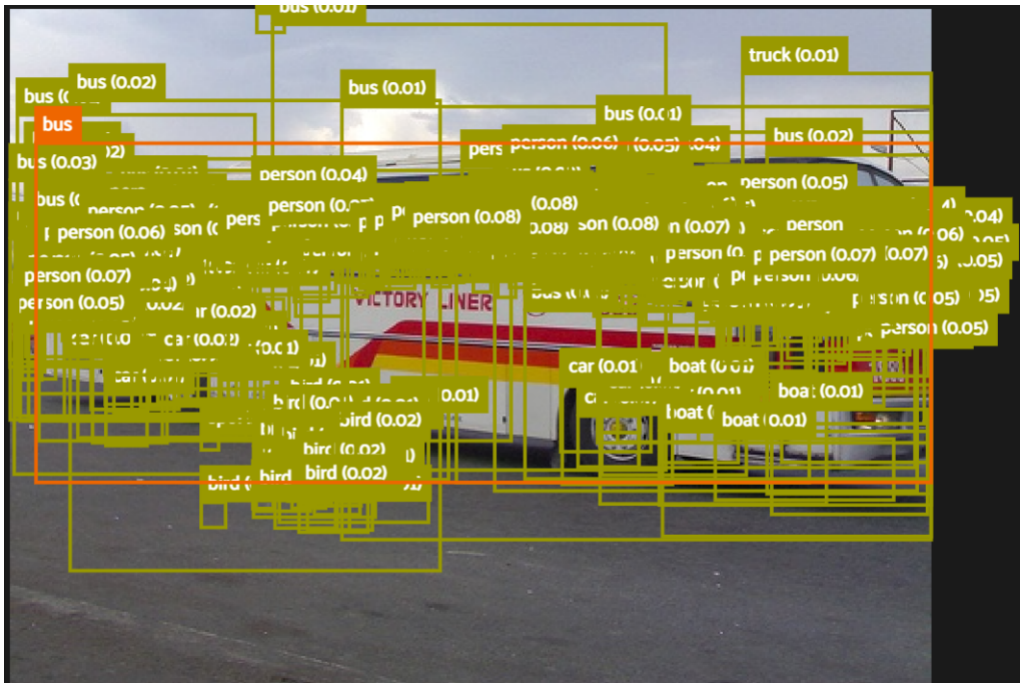
Comments:

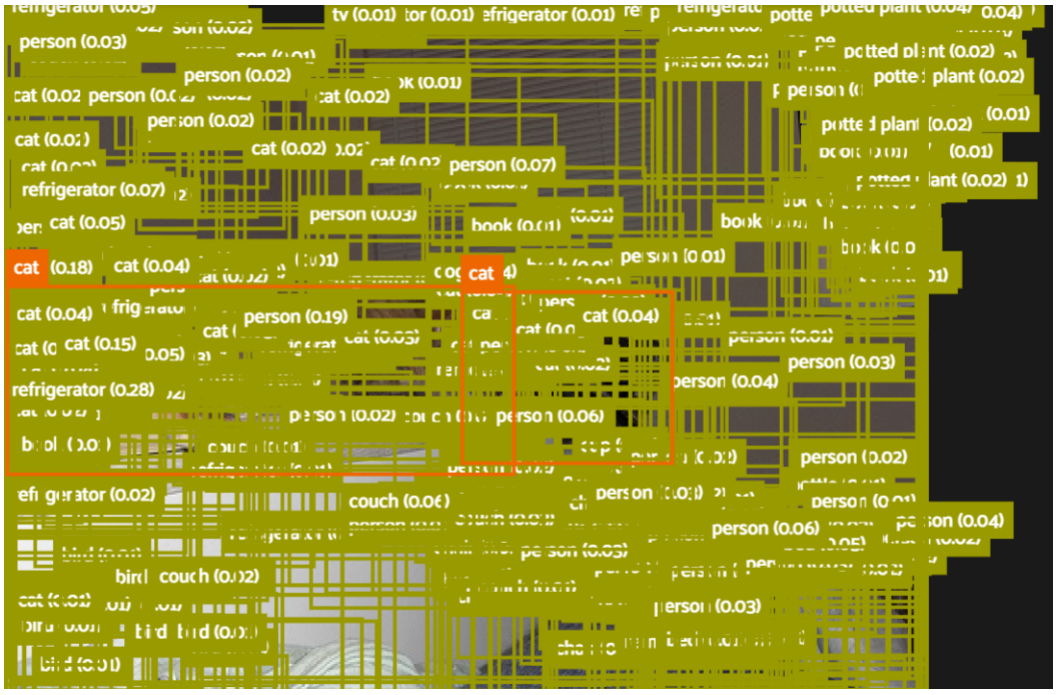
- All these results have confidence of 0.75 or more.

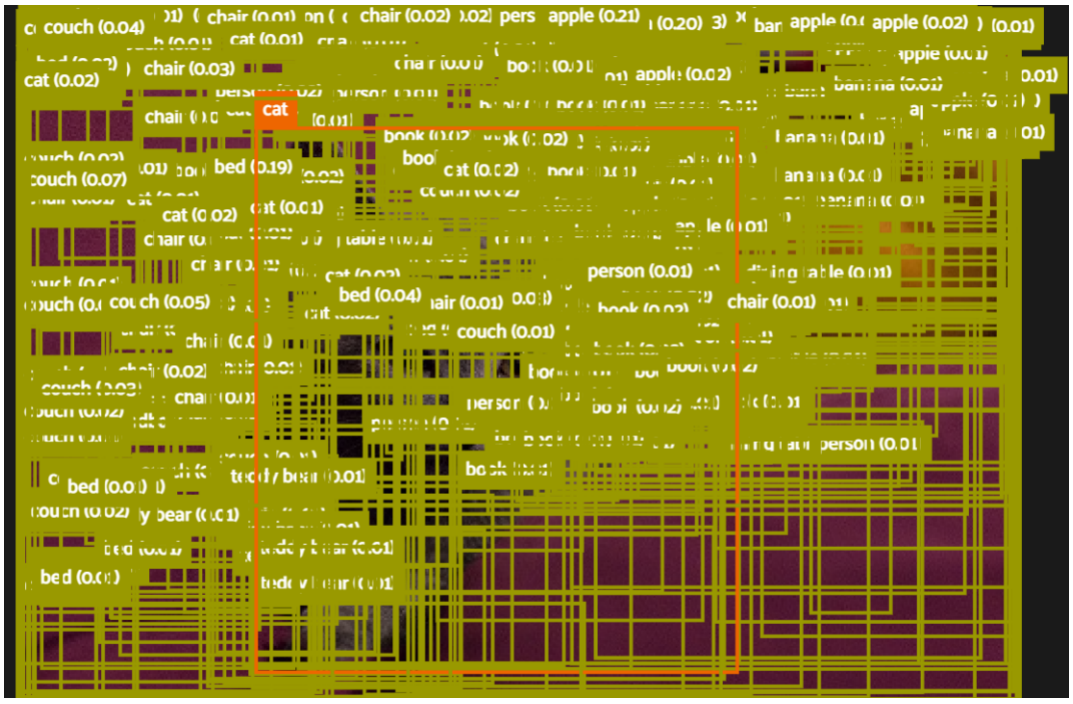
- Same comments COCO dataset.

Failure cases:

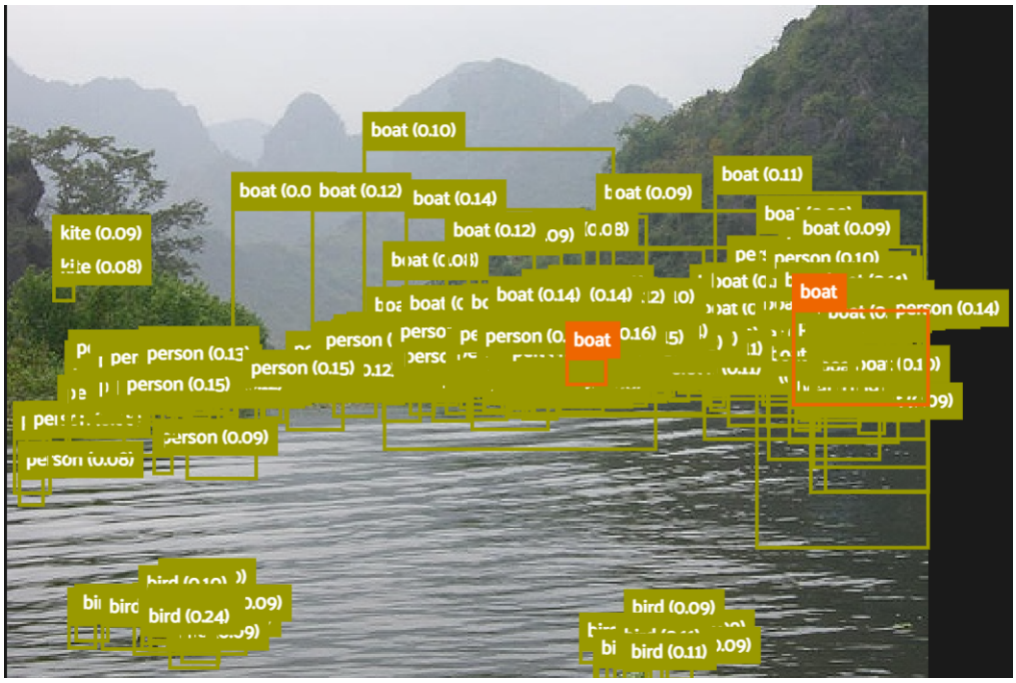



truck (0.01)
bus (0.02)
bus (0.01)
bus
bus (0.02)
person (0.04)
person (0.05)
person (0.06)
person (0.08)
person (0.07)
person (0.05)
car (0.01
boat (0.01)
bird (0.02)


person (0.03)
person (0.02)
cat (0.02
cat (0.02)
person (0.07)
refrigerator (0.07)
cat (0.05)
cat (0.18)
cat (0.04)
person (0.19)
cat (0.15)
refrigerator (0.28)
person (0.06)
couch (0.06)
bird
couch (0.02)
person (0.03)


cat
chair (0.03)
couch (0.07)
bed (0.19)
cat (0.02)
cat (0.01)
couch (0.05)
bed (0.04)
person (0.01)
chair (0.01)
couch (0.01)
apple (0.02)
banana (0.01)


bird
bird (0.05)
bird (0.03)
bird (0.04)
bird (0.09)
bird (0.02)
bird (0.03)
banana (0.02)
banana (0.01)



Comments:

- All these results have confidence of 0.3 or less.

- Same problems as COCO dataset.

5 Analysis

Architecture:

- Faster RCNN ResNet50 FPN and Faster RCNN MobilenetV3 Large 320 FPN are two-stage models which have high localization and recognition accuracy. One model is used to extract regions of objects, and a second model is used to classify and further refine the localization of the object.

- SSDLite320 MobilenetV3 Large is a single-stage model which has high inference speeds.

- Two-stage (multi-stage) models are known to be relatively slow, but very powerful, but recent progress such as sharing features improved two-stage detectors to have similar computational cost with single-stage detectors.

Speed:

- Faster RCNN MobilenetV3 Large 320 FPN showed itself to be the fastest of the three models in both predicting and evaluating the results which is logical because of the small size of the model, these models are considered very useful to be implemented on mobile and embedded devices.

- Faster RCNN ResNet50 FPN was the slowest model in predicting and detecting the objects. However, it took less time than model (3) in evaluating the results.

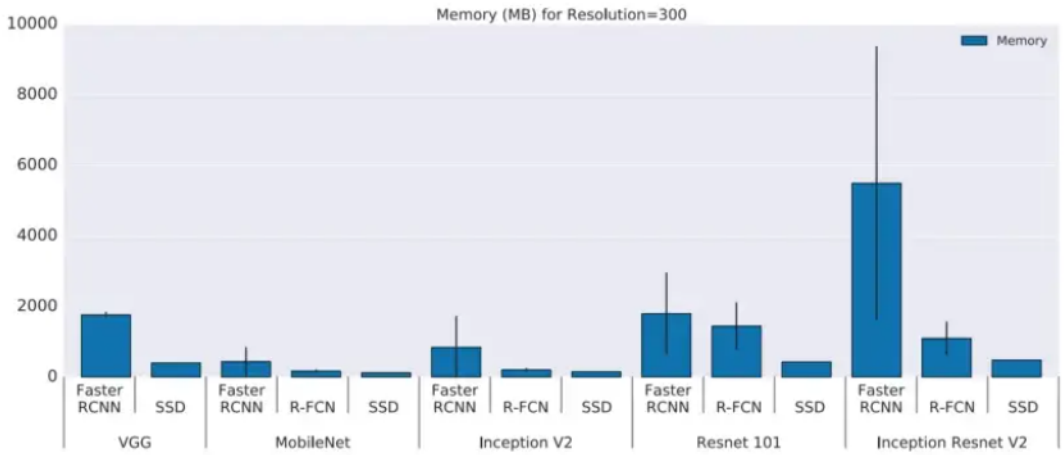
- SSDLite320 MobilenetV3 Large was relatively fast in predicting and detecting the object, but the slowest in evaluating the results because of generating a huge number of bounding boxes.

Memory usage:

- As mentioned before, Mobilenet models are considered very useful to be implemented on mobile and embedded devices which would by default make them require less memory than other models.

- However, Mobilenet models require more memory when used with Faster RCNN than when they're used with SSD.

- The model that needed the most memory was Faster RCNN ResNet50 FPN as it's ResNet with Faster RCNN.



Number of bounding boxes predicted (100 samples):

- Faster RCNN MobilenetV3 Large 320 FPN produces the least number of bounding boxes because it's main aim is to be efficient in both time and space. These boxes were relatively good at detecting the objects, but some objects in the images go undetected.

- SSDLite320 MobilenetV3 Large produced the most number of bounding boxes, but most of them were not accurate and had low confidnce level. SSD usually produces a very large number of bounding boxes for them to be filtered later on either by thresholding or non-maximum suppression to reach the good results.

- Faster RCNN ResNet50 FPN produced an average number of bounding boxes in comparison to the other two models.

Results from testing on the COCO 2017 Dataset:

- Faster RCNN ResNet50 FPN has highest mAP of 0.3693.

- Faster RCNN MobilenetV3 Large 320 FPN and SSDLite320 MobilenetV3 Large have approximately an mAP of 0.22.

- Faster RCNN ResNet50 FPN had the highest number of true positives count

of 630, followed by SSDLite320 MobilenetV3 Large of 468, and finally, Faster RCNN MobilenetV3 Large 320 FPN of 380 over 100 samples.

Generalisation when testing with VOC 2012 Dataset:

- Mean average precision was approximately the same in both datasets, COCO 2017 and VOC 2012 when tested with Faster RCNN ResNet50 FPN.

- Mean average precision increased by approximately the same rate in both datasets when testing with the other two models, Faster RCNN MobilenetV3 Large 320 FPN and SSDLite320 MobilenetV3 Large.

General comparison points:

- Most of the Faster R-CNN models like Faster R-CNN ResNet50 FPN are really great at object detection. But there is one issue. It struggles to detect objects in real-time. Using a mid-range GPU, it is very difficult to get more then 6 or 7 FPS with the ResNet50 backbone. However, MobileNets are really great backbones if you are looking out for object detection in computationally constrained and edge devices.

- Single Shot MultiBox Detector (SSD) is the first single stage detector that matched accuracy of contemporary two stage detectors like Faster R-CNN, while maintaining real time speed.

- It's been noticed that Faster RCNN MobilenetV3 Large 320 FPN is more concerned with precision than recall, this was concluded from it producing a small number of bounding boxes that relatively yield good results.

It's been noticed that SSDLite320 MobilenetV3 Large is more concerned with recall than precision, this was concluded from it producong as many bounding boxes in each image as possible to make sure it can detect as many objects as it possibly can. However, most of these boxes don't indicate actual objects.